Now . . . The Ultimate In Wordprocessing For The Commodore Computer. WORDCRAFT ULTRA!

Wordcraft ULTRA™ turns your Commodore microcomputer into one of the world's most advanced word processing systems, incorporating features previously found only in systems priced thousands of dollars higher!

- True proportional spacing with inter-character and inter-word spacing!
- Multi-user with up to 8 CPU's sharing one or more disk drives and printers with no extra hardware required other than a cable!
- Screen layout matches the printed document! You'll love the "what you see is what you get" feeling.
- Continuous centering, delete and insert, movement of text, search and replace, tab and indent, bold print and underline . . . a seemingly endless list of features designed to make text editing as simple and complete as possible!
- Wordcraft ULTRA runs on the CBM 8032, 8096 or Madison Computer's Z-RAM™ board. The Z-RAM Board not only expands your computer to 96K but also adds CP/M™.

Why settle for less when you get so much more with Wordcraft ULTRA on the Commodore Computer!

Contact Your Nearest Commodore Dealer Today . . .
You'll Be So Glad You Did!

In The East Call:
COMPUTER MARKETING SERVICES INC.
300 W. Marlton Pike, Suite 26
Cherry Hill, New Jersey 08002
(609) 795-9480

In The West Call:
CIMARRON CORPORATION
666 Baker Street, Suite 319
Costa Mesa, California 92626
(714) 641-1156

Z-RAM is a trademark of Madison Computer
CP/M is a trademark of Digital Research
Wordcraft Ultra is a trademark of Dataview Ltd.

Getting The Most Out Of USR

Charles Brannon, Editorial Assistant

The Atari USR command is very powerful and flexible. Its strength is in *parameter passing*, the ability to directly communicate with a machine language routine using standard variables and arithmetic expressions.

A simple task for the USR command is to merely transfer control from BASIC to machine language:

```
X=USR(1536)
```

This would simulate a JSR (Jump to Sub-Routine) to location 1536, or $0600. The value returned in X is meaningless here. The machine language routine must begin with a PLA (Pull Accumulator) to "clear" the count byte (discussed later) and, when finished, return to BASIC with a RTS (ReTurn from Subroutine).

The real power of USR, however, is that it can pass a series of 16-bit binary integers. These are specified as a list after the address:

```
X=USR(1536,1,3,5,7)
```

Any arithmetic expression can be used, even variables and functions:

```
X=USR(1536,A*B,ASC("+"))
```

From the machine language program's point of view, where are these numbers stored? How about the stack? The Atari USR command "pushes" the high and low bytes of each number onto the stack, and "tops it off" with a count byte. The count byte is the number of values passed. The machine language program would use PLA to read each byte into the accumulator. For example, a routine to simulate the Atari POSITION command might look like:

```
; A=USR(1536,X,Y)
*=$600
PLA       ;Count byte
PLA       ;MSB of X
STA 86    ;COLCRS+1
PLA       ;LSB of X
STA 85    ;COLCRS
PLA       ;MSB of Y (zero)
          ;so ignore it
PLA       ;LSB of Y (0-191)
STA 84    ;ROWCRS
RTS       ;Return to BASIC
```

Notice the order of the high byte (MSB) and low byte (LSB) of each argument on the stack. Also, the first argument (X) will be the first value on the stack.

Machine language routines can also work on strings, via the ADR function. ADR(A$) will return the memory location of the contents of A$. Using the LEN function, BASIC can tell the "whole story." For example, this routine transfers the contents of any string to any memory location (useful for player/missile graphics, or custom characters). The length of the string should be limited to 255 bytes.

```
        A=USR(1536,ADR(X$),LEN(X$),MEM)
        *=$0600
ADRL    =$CB
ADRH    =$CC
DESTL   =$CD
DESTH   =$CE
        PLA             ;Count byte
        PLA             ;MSB of address
        STA ADRH        ;zero page loc.
        PLA             ;LSB
        STA ADRL
        PLA             ;MSB of length
                        ;(ignore it)
        PLA             ;LENgth
        TAY             ;Use it for loop
        PLA             ;MSB of destination
                        ;address
        STA DESTH       ;Another z-page loc
        PLA             ;LSB
        STA DESTL
LOOP    LDA (ADRL),Y    ;Get byte
        STA (DESTL),Y
        DEY             ;check loop
        BNE LOOP        ;If not 0,
                        ;continue loop
        RTS             ;Return to BASIC
```

Going Back To BASIC

How can a routine pass a value back to BASIC? It could save the values in an area of memory and have BASIC PEEK them out. If only one value (one 16-bit integer) needs to be returned, you can use locations $D4-$D5 (212,213). Store the result using the standard 6502 low/high byte format. The destination variable (X in X=USR(1536), Z in Z=USR(1536,3,2), or any variable) will take on the value placed in $D4-$D5 (labeled FR0). So, to quickly add two numbers, you could use: A=USR (1536,1,2) (any two arguments). "A" will contain the answer.

```
FR0     =$D4            ;Low byte of return value
        *=$0600
        PLA             ;Throw away count
```

What Is Putting Commodore's 64K Memory Expansion Board Out Of Business?

Z-RAM!

That's right! VisiCalc™, WordPro 5-Plus™, UCSD Pascal™, and special versions of Wordcraft ULTRA and Silicon Office all now run on Madison Computer's Z-RAM Board . . . PLUS you get CP/M™ capability!

What more could you ask for? Except Z-RAM, that is!

Contact Your Nearest Commodore Dealer Today . . .
You'll Be So Glad You Did!

Distributed by:
COMPUTER MARKETING SERVICES INC.

(609) 795-9480
300 W. Marlton Pike, Cherry Hill, New Jersey 08002

Z-RAM is a trademark of Madison Computer
CP/M is a trademark of Digital Research

Teach your Commodore to use the telephone.

Telecommunicate with McTerm

Don't miss out on the computer/telephone revolution. With our McTerm package and a modem, your Commodore becomes an *intelligent* terminal. You can interact with large and small computers anywhere in the world.

Access databases like the Source™, or MicroNet™, or Dow Jones for up-to-the minute news, sports, stock market reports, etc. Write programs on your micro and run them on distant mainframes.

McTerm actually saves you money and time while you use it. There's no need to read data line-by-line as you receive it. Instead, you can quickly save it all to disk or printer and sign-off. You save on expensive computer time *and* long distance phone rates.

McTerm sends and receives faster than anything else on the market – up to 1200 baud. Adjustable duplex, echo, and parity. Runs on any Commodore 4000 or 8000 series computer with BASIC after version 1.0. Works with most RS-232 modems and supports auto dial. Write/Or Phone for Data Sheet.

MC

McTerm connects you with the rest of the world.

Madison Computer, 1825 Monroe, Madison, WI 53711 USA. 608-255-5552

```
        PLA
        STA FIRSTH
        PLA
        STA FIRSTL
        PLA
        STA SECONDH
        PLA
        STA SECONDL
        CLC
        LDA FIRSTL
        ADC SECONDL
        STA FR0
        LDA FIRSTH
        ADC SECONDH
        STA FR0+1
        RTS
```

In many programs, we want to make sure that the proper number of arguments has been sent. For example, if we have a routine that plays a musical tone on the internal speaker for a specified duration,

```
A=USR(1536,note,duration)
```

we may want to only accept exactly two values. We can use the first byte, the count byte, to monitor this. If the count is wrong, we must pull all the arguments off the stack and return to BASIC. We could even ring the bell and print an error message.

```
        *=$0600
        PLA
        CMP #2
        BNE ERROR
        .
        .
        (Routine continues normally)
        .
        .
        .
        RTS
ERROR                       ;The error-handling
                            ;routine
        TAX                 ;Count is in A
        BEQ NOPULL          ;If zero, don't pull
ERRLOOP                     ;ERROR loop
        PLA
        PLA                 ;Pop an argument
        DEX                 ;Continue
        BNE ERRLOOP         ;Until X=0
NOPULL
        LDA #253            ;BELL character
        JSR $F6A4           ;Print it
        LDA #03             ;ERROR - 3
                            ;(VALUE ERROR)
        STA $B9             ;Error number
        JMP $B940           ;Print error
```

Machine language programmers have a friend in USR. If you have an Assembler, type in the examples. And when BASIC bogs you down, remember this motto: Use USR! ©

Review:

VisiCalc Home And Office Companion

Terry Nelson
Camas Valley, OR

The paragraph began, "If your computer can run the VisiCalc program, you can enter and use any of these 50 models immediately."

I thought: "All I need is another book of useless, simplistic cookbook routines that have no application to real life!"

My past experiences with books of "popular routines in ABC BASIC" were far from satisfying. Their dusty covers are a constant reminder to me of how not to spend my money. I soon discovered, however, that the *VisiCalc Home and Office Companion* (henceforth VHOC) was not quite what I had expected, both in the type of applications addressed and in their usefulness to my business.

The Approach

For each of the 50 applications a VisiCalc spread sheet is shown, along with a listing of the VisiCalc instructions which will produce it. For instance, in the "Grade Book" model, a spread sheet is shown formatted to resemble a typical grade book. Along the left column are the students' names, and to the right are columns titled "SCORE FOR TEST 1, SCORE FOR TEST 2," etc. On the far right is a column which reports each student's test average. The last row of the page displays the class average for each of the tests. If you boot VisiCalc and type in the listing, the blank worksheet will be formatted to resemble this grade book. Once you have typed the listing in, you just change the names and scores to your class data, save the file, and the grade book is ready to use. Each time you want to look at it you boot VisiCalc, enter the storage command: /SL GRADE BOOK, and the book is displayed on the screen. Since the book is a VisiCalc file, any time you enter a new name or change a score or put in a new column, you do it through VisiCalc.

The listing for each model is easy to follow. VisiCalc commands and data are entered by column; therefore, column A is entered first, then column B, etc. The commands are mixed with the data, so care must be taken to follow the listing exactly. For instance, portions of the listing for the D column in "Sales Commission Register" look like this:

```
>D 1:"MISSIONS
>D 4:"     (SALES
>D 6:/FL"   TO
>D 7:1
>D 8:3001
 ........
>D29:@SUM(D23...D27)
```

The Selection

The VHOC contains a number of useful models which enable the user to quickly set up meaningful, attractive VisiCalc files. The models are organized into seven categories: loans and investments, general business, inventory control, advertising and sales, personnel and departments, personal finance, and household aids. I was pleasantly surprised to find the majority of the models devoted to business and finance. It's hard for me to get excited about recipe conversion programs, tire rotation reminders and other so-called household computer applications. There are certainly home applications for computers, but they're a bit more substantive than these, I hope. Of the three models presented in the household aids section, "Events Scheduling" is trivial. "Paint a Room" is too complicated for a simple room painting job.

Out of the 50 models presented, about 20 of them are generally helpful. These make effective use of VisiCalc's features. The others either tend to be directed to a small audience or are more trouble to set up than they are worth in terms of the jobs they perform. "Business Startup Worksheet" might be helpful to someone who regularly starts businesses, but if and when I ever start another business I won't spend an hour typing in a model when I can jot down the same expense categories and total them in 15 minutes. Similarly, I'm not interested in "Travel Log." My little auto record book in the glove compartment keeps sufficient records of my trips without my having to type in all that information again at the computer just to get it neatly categorized on a printout.

I will probably use several of the better models in my own business. "Professional Service Fee Analysis" is very helpful for setting reasonable fee rates for your own consulting business. The "Sales Commission Register" elegantly calculates sales commissions on a sliding scale and keeps a running year-to-date tally on both commissions and draws. If you've been wondering what your net worth is, "Net Worth Statement" will remind you of the important assets and liabilities to consider and then assemble them into a convenient report. There are stock and bond portfolios, a rental property evaluator, a cash flow analyzer and various financial

YOU ARE GOING TO SMILE!

--When you see over 2000 books, programs, and accessories we carry for All Major Brands in our Computer 'SOURCE BOOK®'. From introductory level to professional, we cover it all. Listed Below is just a sample of what we carry.
DEALER INQUIRIES WELCOME

JMC

INTRODUCTORY BOOKS

Introduction to Word Processing. Plain language book for those considering a purchase.
4795-000076 300 pgs,140 illus $12.95

How to Buy the Right Small Business Computer. Easy to follow book with 'checklist'.
4925-008494 225 pgs, $ 8.95

Business System Buyers Guide. This guide cuts thru the jargon and simplifies your task.
4665-000047 166 pgs * $ 7.95

Using Micros in Business. Essential background briefing for any buyer of hard.or software
4410-005152 192 pgs $ 9.95

Your Home Computer. Meant for the person who has no technical background,-buying tips!
4250-000022 211 pgs $12.95

Home Computers:Beginners Glossery & Guide. Terminology,number systems,symbols
4250-000002 147 pgs, 20 illus. $10.95

Why do YOU need a Personal Computer? Learn the 100's of everyday applications.
4925-004784 320 pages $ 8.95

Computer Dictionary & Handbook. More than 22,000 definitions,acronyms,abbreviations.
4760-021632 928 pages $34.95

Computer Dictionary. More than 12,000 definitions clear up almost any question you have.
4760-021652 624 pages, $15.95

How to Computerize Your Small Business. Shows how to plan a tailored computer system
4690-403857 171 pages $7.95

How to make MONEY with Your Micro. This book gives you dozens of money making ideas.
4250-000074 154 pages $12.95

Playing the Stock & Bond Markets w/ Micros. Learn the principals of investing in the markets
4800-001251 308 pgs,35 illus $ 9.95

GENERAL INTEREST BOOKS

Background Math for a Computer World. All the basic math facts,techniques,& concepts
4925-008086 308 pages $ 7.95

Micro's and the 3R's. This book educates the educator on ways to teach with computers.
4410-005113 192 pages $ 9.75

Intro to Computer Music. The book explains in detail computer music, non-technical 'how to'.
4925-086839 313 pages $14.95

1001 Things to do with your Personal Comp Time saving,money saving & just plain fun ideas
4800-000160 336 pages $ 7.95

Computer Graphics Primer. Only book on the market that shows how to do comp. animation!
4760-021650 184 pgs, $14.95

How to Troubleshoot and Repair Micro's. How to locate & repair common problems.
4725-002981 290 pages $ 7.95

Visicalc™: Home & Office Companion. A compendium of 50 VisiCalc™ models & samples
4665-000050 182 pgs,12 illus $15.99

Wordstar™ Made Easy. In 14 east lessons this handy book will teach you the powerful features
5664-000069 125 pages $11.95

ATARI™ BOOKS

rs030A Guide to Atari™ 400/800 Computers Provides invaluable material for Atari™ users.
4665-000065 458 pages $16.95

PAINT. Create 'paintings' of your own on the 400/800 system. Both beginning and pro levels
4725-005425 $29.95

The Atari™ Assembler. Master assembly language,you will need some BASIC experience.
4690-000236 270 pages $ 9.95

Atari™ Games & Recreations. Provides beginning or advanced users with games & 'how to'.
4690-000242 252 pgs,225 illus $12.95

Computers For People. This book outlines the benefits and uses of Atari™ 400 or

Computers for People. This book outlines the benefits and uses for Atari™ 400 & 800 systems
4250-000064 200 pgs, 47 illus. $ 7.95

Atari™ Basic. A fine intro to BASIC programming. Turn from novice to pro in week or so.
4925-006496 $ 8.95

Atari™ Sound & Graphics. This is a self-paced book to fully explore the expressive potentials.
4925-009593 160 pages $ 9.95

The VisiCalc™ Book: Atari™ Edition. Make the most of VisiCalc™ capabilities on the Atari™.
4725-008393 400 pages $14.95

Inside Atari™ DOS. Step by Step guide to the DOS software. Complete with source code
4105-000002 120 pges $19.95

GOT SOMETHING YOU WANT TO SELL?
JMC is always looking for new items to add to our Computer 'Source Book'. If you have a book, or software for one of the popular personal/home computer systems please write to us with details. Perhaps we can add it to the thousands of items we are currently selling to computer retailers all over the USA

If you have a product you would like to market -but don't want all the hassle of packaging, promoting,and producing the product---let us know, perhaps we can work out a generous Royalty arrangement with you.

TRS-80™ COLOR COMPUTER BOOKS

55 Color Computer Programs for the Home, School & Office. Includes lots of color graphics
4080-000005 128 pgs $ 9.95

55 More Color Computer Programs. Companion volume to above,more useful programs.
4080-000008 112 pgs $ 9.95

Love Poems from the Digital Heart of an electronic Computer. An ideal Valentine!
4080-000013 96 pgs $ 4.95

TRS-80™ Color Computer Graphics. Comprehensive coverage of color and graphics avail.
4725-007864 $14.95

Color Computer Graphics. A must have book for owners of the TRS-80™ Color Computer.
4080-000012 128 pgs $ 9.95

Color Computer Songbook. Book of favorite songs and classics to be played on Color Comp.
4080-000011 96 pgs $ 7.95

TRS-80™ Color Computer Applications. A beginners handbook loaded with examples.
4690-007870 240 pgs $12.95

TRS-80™ Color BASIC. Self-teaching guide leads novices step by step into programming

TRS-80™Color BASIC. Self-teaching guide leads novices step by step into programming.
4925-009644 256 pgs $ 9.95

TRS-80™ Color Programs. 37 programs are fully documented in this easy to follow book.
4250-000061 323 pgs $19.95

101 Color Computer Programming Tips & Tricks. Practical,useful techniques & shortcuts
4080-000007 128 pgs $ 7.95

LANGUAGE BOOKS

Introduction to BASIC. Extremely popular introduction to BASIC,emphasis on fundamentals
4515-000022 206 pgs,20 illus $10.95

50 More Programs in Basic for the Home, School & Office. All fresh programs in this vol.
4080-000003 96 pgs $ 9.95

BASIC Programming Primer. User-friendly fundamentals book for both novices and pros.
4760-021586 240 pgs $11.95

Sixty Challenging Problems with BASIC solution. Learn more by solving problems.
4410-005180 144 pgs $ 8.50

Beat the Odds. Micro simulation of Casino Games. Painless way to gamble your money
4410-005181 128 pgs $ 8.70

Introduction to Texas Instrument BASIC. A must for any TI™ owner.
4410-005185 320 pgs $11.95

BASIC for Home Computers. A sensible and complete beginning guide to BASIC.
4925-003204 336 pgs $ 9.50

BASIC,Vol II. Picks up where above book left off,needs no special math or science knowledge
4925-003500 325 pgs $ 9.95

Inside BASIC games. Teaches the reader how to design error-free interative games,etc.
4795-000055 350 pgs,120 illus $14.95

BASIC Programs for Home Financial Management. 33 Useful programs for you.
4690-066514 250 pgs, illus $12.95

BASIC-BASIC-English Dictionary. This book will help you translate between BASIC dialects
4250-000054 150 pgs $10.95

Basic Programming Worksheets. Printed worksheet forms,40 sheets to pad. 80 columns
4080-000802 For Color Computer $2.95
4080-000801 For Pocket BASIC $2.95
4080-000803 For Apple™ $2.95
4080-000804 For IBM™ PC $ 2.95
4080-000805 Universal BASIC form $ 2.95

BASIC for Business. Shows how easy it is to use a computer for general business tasks.
4795-000080 250 pgs $12.95

A to Z Book of Computer Games. Here are 26 exciting and instructive BASIC game programs
4800-001062 308 pgs,73 illus. $ 7.95

40 Computer Games. Unbeatable game collection from Kilobaud Micro Magazine.
4910-000023 $7.95

The BASIC conversions Handbook For Pet, Apple™,TRS-80™ users. Guide to converting.
4410-005534 80 pgs $ 7.95

SINCLAIR ZX-81™ BOOKS

Programming the ZX-81 for Real Applications. Practical programs to show ability of ZX-81
4250-000090 176 pages $ 9.95

Making the Most of Your ZX-81. Information on how to write programs games,tricks,etc.
4725-004188 128 pages $10.95

The ZX-81 Pocket Book. Handy programming manual that gives you everything you need.
4725-009524 128 pages $9.95

Mastering Machine Code on Your ZX-81. First 'simple english' guide to machine code.
4725-004261 180 pages $12.95

49 Explosive Games for the ZX-81. Complete easy to read game rules and program instruct.
4725-002086 140 pages $10.95

APPLE™ BOOKS

Apple II™ Users Guide. This guide is the key to unlocking the full power of your Apple II.
4665-000046 385 pgs $16.95

APPLE™! Collection of BASIC programs that have been converted to run on the Apple II.
4665-000068 200 pages. $14.99

Assembly Lanquage Programming for the Apple II™. Comprehensive introduction.
4665-000051 $12.95

Apple™ Machine Language. A machine language book that's fun and entertaining.
4690-000230 296 pages/ illus. $12.95

Apple™ Interfacing. Helps you to interface your Apple™ to a variety of electronic devices.
4760-021862 208 pgs $10.95

Apple™ Programming Exercies. 17 Easy to read,easy to use exercies including music.
4925-086598 192 pages $ 9.95

AppleSoft™ Language. Complete text on the AppleSoft™ BASIC, easy to understand book.
4760-021811 256 pages $10.95

33 New Apple™ Programs for Home, School & Office. Minor changes for other systems too
4080-000016 $ 8.95

101 Apple™ Programming Tips & Tricks. Thoroughly tested hints,secrets techniques
4080-000015 128 pages $ 8.95

Golden Delicious Games for the Apple™. Book that lets readers create their own games.
4925-009083 256 pages $12.95

Apple™ Word Processing. Independent and objective evaluation of Apple™ word processor
4700-000005 250 pgs $19.95

The VisiCalc™ Book: Apple™ Edition. Guide to using VisiCalc™ productively on Apple II™.
4725-008397 400 pages $14.95

Pie Writer. One of the fastest,most efficient & economical word processing systems on the market,great documentation,upper & lower case,incremental spacing and more! Needs 48K
4415-012009 40 column version $149.95
4415-013409 80 column version $149.95

Apple™Computer for Beginners. For the uninitiated,offers hands-on teaching guide.
4690-039131 304 pgs, due Fall 82 $12.95

PROGRAMABLE CALCULATOR BOOKS

Pocket Computer Programming Made Easy Quick Easy way to make your computer perform.
4080-000009 128 pages $ 8.95

35 Practical Programs for the Casio PC. For office,home,school..ready to run programs.
4080-000014 96 pgs $ 8.95

Murder in the Mansion and other adventures You will have fun with this assortment of games
4080-000501 96 pages $ 6.95

Calculators User Guide & Dictionary. Books purpose to minimize confusion between brands
4515-000006 425 pages $12.95

Games,Tricks,& Puzzles for a hand Calcu. Jam packed with useful and fun information.
4250-000008 91 pages,30 illus $ 4.95

Programmable Calc.:How to Use Them. The 'how to' of programming and compares brands.
4515-000008 526 pages 123 illus $15.95

Computer 'Source Book'™. Over 2000 books programs,and accessories for all micro brands
0001-198207 72 pgs $2.00 ppd.

PROTECT YOUR INVESTMENT

Your Personal Computer is an expensive investment,if taken care of it will give you hours of enjoyment and operate trouble free for years. BUT, accidents due happen, and we suggest you take out some cheap insurance to protect it. We are pleased to offer you these hi-quality vinyl covers all are Saddle Tan in a leatherette finish.

4108-000001 Vic-20™ Keyboard $10.98
4108-000002 Vic-20™ Datasette $ 4.98
4108-000003 Vic 1515 printer $10.98
4108-000004 Vic 1540 Disk Drive $ 9.98
4108-000005 Atati™ 400 $11.98
4108-000005 Atari™ 400 $11.98
4108-000006 Atari™ 410 $ 4.98
4108-000007 Atari™ 800 $12.98
4108-000008 Atari™ 810 $12.98
4108-000009 Atari™ 820 $ 8.98
4108-000010 Atari™ 822 $ 7.98
4108-000011 Atari 825 $11.98
4108-000012 Atari 825(w/paper roll)$11.98
4108-000013 Apple III™ $15.98
4108-000014 Apple II™ $15.98
4108-000015 Apple II™ Keyboard $ 9.98
4108-000016 Apple II Single Disk $ 5.98
4108-000017 Apple II Disk(2 stacked)$ 9.98
4108-000020 Texas-Instru. 99/4™ $10.98
4108-000026 TRS-80™ Color Comp. $11.98

PET/CBM BOOKS

Pet™ Pesonal Computer Guide. Discusses all the 4000 and 2000 series, plus new Pet™ Items
4665-000076 500 pgs,illus $15.00

Pet Graphics. Booklet willhelp you use one of the most powerful features of the Pet/CBM.
4840-000002 $ 4.95

32 Basic Programs for the Pet. Chock full of practical applications,ed uses,games & graphics
4250-000025 267 pgs,56 illus $19.95

PET fun and Games. 30 challenging games for PET computers,complete details & instruct.
4665-000070 170 pgs $11.95

PET™ Interfacing. Demonstrates how you can build numerous devices for interfacing to a PET.
4760-021795 264 pgs. $16.95

VIC-20™ BOOKS

Vic-20™ Programmers Ref. Guide. A must for any Vic owner,detailed easy to use manual.
4100-000110 Approx 400 pgs $16.95

Understanding Your Vic.Vol 1. Beginners guide to programing on the Vic-20™. Tips & tech.
4840-000003 148 pages $11.95
4840-000004 Cassette Supplement $7.95

Compute!'s First Book of Vic. Collection of the best articles from Compute! Magazine.
4105-000007 Fall 1982 $12.95

Starting with BASIC on the Vic-20™.Helps novices learn all the capabilities of the Vic-20
4725-007070 128 pgs,due Fall 82 $12.95

IBM™ BOOKS

IBM's™ Personal Computer. Background on the giants entry into field,with evaluations.
4700-000111 303 pages $14.95

Using the IBM™ Personal Computer. A complete handbook showing what it can do.
4725-008183 300 pages $12.95

CONFUSED? No one book may offer all the answers,we suggest buying more than one book on a subject to get as many different views and opinions as possible. Books are cheap if you consider the frustrations they can avert.

FREE

--With ANY order of $50.00 or more you can receive a 1983 Computer Calendar FREE,--A $ 7.95 VALUE!

COUPON

QTY	NUMBER	DESCRIPTION OF ITEM	COST	TOTAL

(Orders Under $25 Please Add $2.50 Pcst & Handling -- Illinois Residents Please Add 5% Sales Tax. ALL CATALOGS SHIPPED POST PAID) TOTAL

PAYMENT ENCLOSED: ☐CASH ☐CHECK ☐MONEY ORDER

PLEASE CHARGE TO MY: ☐MASTERCARD ☐VISA (Min. Chg. $25.)

CARD NUMBER________

EXPIRES________ INTRBNK #________

SHIP TO:________

STREET ADDR:________

CITY________ STATE______ ZIP______

JMC

DEPARTMENT: G-2
1025 INDUSTRIAL DR., BENSENVILLE, IL 60106-1297

PRICES AND AVAILABILITY SUBJECT TO CHANGE schedules for retail and manufacturing businesses. Basic accounts receivable and payroll and inventory control models are presented as well as several project schedules.

A word of caution is appropriate here, I think. If I were in the market for a bookkeeping system, I would look for one with as little damage potential as possible. All of these models fail in this regard since the systems themselves can be easily modified by the same VisiCalc commands that are being used to update data. In conventional computer bookkeeping systems, the only way the operator can mess the results up is by entering erroneous numbers. In a VisiCalc based system, a few simple one-button commands can wipe out entire *columns* of data and programmed commands. Not only that, but a few more one-button commands will wipe out the original file! These are serious limitations. Office environment pressures are often intense, and, if careful concentration must be maintained to operate a system, there will come a day when it's not maintained and the results can be disastrous. If you plan to use any of these models for bookkeeping, train the operator carefully, make periodic file printouts, hide a backup disk for your own peace of mind and provide the operator with a library atmosphere to work in.

The documentation supplied with each application, in general, is sufficient to explain ambiguous data titles and operation procedures; however, the VHOC is not a business or investment primer. The "Mini Accounts Receivable" model is a workable ledger, but don't expect to learn bookkeeping procedures for accounts receivable from the half-page of documentation given with it.

The majority of the models presented in the VHOC are useful and practical helps for investment analysis and business planning. Every VisiCalc user could probably apply at least two or three of these. The models themselves are excellent examples of how to format the VisiCalc worksheet to print professional looking expense reports and balance sheets. With careful consideration of the limitations inherent in the VisiCalc "operating system," several of these models can be used effectively for bookkeeping. The authors have effectively shown how to use VisiCalc in applications for which I never would have considered VisiCalc. If none of the models had been useful to me, the book would still have been a valuable purchase for that education alone.

VisiCalc:
Home and Office Companion
Osborne/McGraw Hill
$15.99 paperback
182 pages

©

Review:

Speech Synthesizers For Atari And Apple

Charles Brannon, Editorial Assistant
Tom R. Halfhill, Features Editor

Let's be honest. How many of us have watched *Star Trek* on TV without wishing that *our* computers could talk, too?

Synthesized speech has been around for a few years now, but has cost hundreds of dollars, even on microcomputers. That's why two new speech products for the Atari and Apple have stirred so much interest – their quality sets a new high, their prices a new low.

The Alien Group's new Voice Box ranges from $139 to $215, while Don't Ask Computer Software's revolutionary synthesizer on a disk – *Software Automatic Mouth* (*S.A.M.*) – checks in for even less at $59.95 (Atari version) and $124.95 (Apple version). Both are capable of startlingly human-like speech.

The two products approach the problems of speech synthesis in quite different ways, however. The Voice Box is a plug-in "little black box" supported by machine language programs that allow you to create and store dictionaries of frequently used words. *S.A.M.*, however, is entirely software-based, using no hardware at all (except for a simple digital-to-analog converter and amplifier board in the Apple version).

Since both products hit the market at almost the same instant, and since both are for two of the most popular personal computer systems, there's bound to be brisk competition as people line up on each side of the which-is-best fence. Therefore, we'll state up front that neither will be declared the clear-cut victor here; both are good products, and each has its strengths and weaknesses.

Keeping that in mind, we can explore several criteria for evaluating microcomputer-based speech synthesizers. These include speech quality, versatility, and the ease of incorporating speech into user-written programs.

Is It Human?

Speech quality is probably the most important of these. How closely does the synthesized speech simulate human speech? Both *S.A.M.* and the Voice Box speak in recognizable tones which approach human speech very closely. Both voices are male, not because the programmers were sexist, but because female voices are harder to synthesize due to their wider dynamic range.

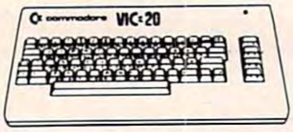
S.A.M. speaks with a definite accent, although the nationality is hard to place. To some it sounds somewhat Scandinavian, perhaps Swedish. Then again, it might be East European. At any rate, *S.A.M.* speaks English as if it were a second language. This is not intended as criticism; on the contrary, *S.A.M.* talks very brightly, enunciating words and syllables with a sense for inflection and accent that is quite amusing. Some syllables sound sort of thick or fuzzy (especially a "th"), as *S.A.M.* struggles to do with silicon chips what a person does with a tongue and palate.

The Voice Box is distinctly different. It speaks in a smoother voice than *S.A.M.*, without as many fuzzy syllables, although it, too, has trouble with certain sounds (a "g" resembles a "d"). However, the Voice Box tends to speak in a monotone when converting plain English to speech, while *S.A.M.* adds its own unique intonation. If the Voice Box speaks with any accent at all, it is "computerese": neutral, unemotional. The nuances are hard to describe, but the results are that the Voice Box tends to offer the more human-like *tones*, while *S.A.M.* tends toward more human-like speech *patterns*.

To put this another way, if you were to have each synthesizer read a plain English sentence over the telephone to a person who was unaware that a computer was speaking, the Voice Box would be quickly identified as a computer, while *S.A.M.* might more easily pass as a human, albeit one with a heavy foreign accent.

Remember, though, we're talking about each product's ability to interpret plain English. English is a formidable challenge because it is a language of as many exceptions as rules. To program a computer with a complete knowledge of English pronunciation – to distinguish between *though, bough,* and *tough,* for example, would require massive amounts of time and memory.

Considering this difficulty, *S.A.M.*'s text-to-speech "Reciter" program works surprisingly well. Given ordinary English text, the Reciter will pronounce it, even adding inflection automatically. The Voice Box uses a "dictionary" to memorize words you "teach" it. If it learns many common patterns such as "ch", "ou", etc., it can simulate a simple text-to-speech algorithm. The advantage of



commodore

VIC 20 . . . . . $179

VIC 1530 Datasette	$ 59
VIC 1540 Disk Drive	$299
VIC 1525 Graphics Printer	$309
VIC 1210 3K Memory Expander	$ 34
VIC 1110 8K Memory Expander	$ 52
VIC 1111 16K Memory Expander	$ 92
VIC 1011 RS 232 Terminal Interface	$ 43
VIC 1211 Super Expander	$ 52
VIC 1212 Programmers Aid Cartridge	$ 45
VIC 1213 Vicmon Machine Language Monitor	$ 45
VL 102 Introduction to Basic Programming	$ 21
VT 106A Recreation Pack	$ 45
VT 107A Home Calculation Pack	$ 45
VT 164 Programmable Character Set	$ 12
VIC 1600 Vicmodem	$ 89
VIC 1311 Joystick	$ 8
VIC 1312 Game Paddles	$ 16
VM Programmers Reference Guide	$ 14

VIC Software

Avenger	$ 23
Superslot	$ 23
Super Alien	$ 23
Jupiter Lander	$ 23
Draw Poker	$ 23
Midnight Drive	$ 23
Radar Rat Race	$ 23
Raid on Fort Knox	$ 23
Sargon II Chess	$ 29
Super Smash	$ 23
Cosmic Cruncher	$ 23
Gorf	$ 29
Omega Race	$ 29
Money Wars	$ 23
Menagerie	$ 23
Cosmic Jailbreak	$ 23
Clowns	$ 23
Garden Wars	$ 23
Sea Wolf	$ 23
Adventureland	$ 29
Pirate Cove	$ 29
Mission Impossible	$ 29
The Count	$ 29
Voodoo Castle	$ 29
The Sky is Falling	$ 23
Mole Attack	$ 23
Bingo Speed Math	$ 23
Home Babysitter	$ 23
Visible Solar System	$ 23
Personal Finance	$ 29

United Microware

Spiders of Mars	$ 34
Meteor Run	$ 34
Amok (C)	$ 17
Alien Blitz (C)	$ 17
Skymath	$ 12
Space division	$ 12
Super Hangman	$ 14
The Alien	$ 17
3D Maze	$ 12
Kosmic Kamikaze	$ 17
Sub Chase	$ 17
Amok (CT)	$ 27
Renaissance	$ 34
Alien Blitz (CT)	$ 27
Cloud Burst	$ 27
Satellites and Meteorites	$ 34
Outworld	$ 34

Creative Software

Black Hole	$ 36
Trashman	$ 36
Astroblitz	$ 36
City Bomber & Minefield	$ 20
Apple Panic	$ 36
Choplifter	$ 36
Serpentine	$ 36

American Peripherals

Snakman	$ 18
Astrobase-2001	$ 10
Victoroids	$ 25
Intruder-Scramble	$ 18
Mobile Attack	$ 25
Crack	$ 25
Guardian	$ 25
Quark	$ 25

FRIENDLY SERVICE

APPLE SOFTWARE NEW

Lazer Maze	$ 20
Federation	$ 20
Shootout at the O.K. Galaxy	$ 19
Sea Fox	$ 20
Space Kadet	$ 24
Mars Cars	$ 20
Money Muncher	$ 20
Vortex	$ 20
Pandora's Box	$ 20
High Orbit	$ 20
Kamikaze	$ 23
Shuttle Intercept	$ 23
Free Fall	$ 20
Wayout	$ 27
The Blade of Blackpool	$ 27
Ali Baba & the Forty Thieves	$ 22
Guadacanal	$ 39
The Cosmic Balance	$ 27
Galactic Gladiators	$ 27
Juggles Rainbow	$ 30
Bumble Games	$ 39
Bumble Plot	$ 39
Gertrude's Secrets	$ 49
Gertrude's Puzzles	$ 49
Rocky's Boots	$ 49
Pest Patrol	$ 20
Lunar Leeper	$ 20
Cannonball Blitz	$ 23
Laffpak	$ 23
Frazzle	$ 17
Firebug	$ 17
S.E.U.I.S.	$ 27
Snooper Troops #1	$ 30
Snooper Troops #2	$ 30
Story Machine	$ 23
Face Maker	$ 23
Lazer Silk	$ 20
Zenith	$ 23
Phaser Fire	$ 20
Prisoner II	$ 22

COMPUTER OUTLET

*** SPECIALS OF THE MONTH ***

Slagh 24K Memory Board — VIC 20	**$ 145**
Slagh 6-Slot Expansion Mother Board	**$ 119**
Smartmodem II	**$ 269**
Amdex Color II Monitor	**$ 759**
NEC 3510 Printer	**$1699**
Microline 84 Printer	**$1089**
The Grappler Plus	**$ 139**
Novation Apple Cat II Modem	**$ 329**
Verbatim Disks (Box)	**$ 27**
Anadex Printer	**$1289**
Hayes Smartmodem 1200	**$ 539**
Hayes Chronograph	**$ 199**
System Saver Fan	**$ 65**
The Joyport	**$ 53**
Wico Trackball	**$ 49**
Wico Joystick	**$ 23**
Wico Joystick Deluxe	**$ 26**
Wico Famous Red Ball Joystick	**$ 24**

Computer Outlet

Park Place — Upper Level
1095 E. Twain — (702) 796-0296
Las Vegas, Nevada 89109

Call Toll Free **800-634-6766**
We accept Major Credit Cards
Mon.-Fri. 8 A.M.-6 P.M.
Sat. 9 A.M.-5 P.M.

APPLE SPECIALS

Entertainment

Bez-Wars	$ 19
Bez-Man	$ 15
Baseball	$ 20
Peeping Tom	$ 23
Fly Wars	$ 20
Kabul Spy	$ 23
The Gauntlet	$ 24
Sherwood Forest	$ 23
The Queen of Phobos	$ 23
Zero Gravity Pinball	$ 20
Amoeba Man	$ 24
Marauder	$ 24
Knight of Diamonds	$ 24
Cytron Masters	$ 27
Star Blazer	$ 22
Swashbuckler	$ 23
Twerps	$ 20
Minotaur	$ 23
Lemmings	$ 20
Bandits	$ 23
Dueling Digits	$ 20
Cyclod	$ 20
Frogger	$ 23
Choplifter	$ 23
Serpentine	$ 23
David's Midnight Magic	$ 23
Escape (A2-SG1)	$ 22
Congo	$ 23
Goldrush	$ 23
Pig Pen	$ 20
Tunnel Terror	$ 20
Threshold	$ 27
Time Zone	$ 65
Ulysses & The Golden Fleece	$ 23
Gorgon	$ 27
Ceiling Zero	$ 20
Raster Blaster	$ 20

Business

Screenwriter II	$ 89
Visicalc 3.3	$ 179
The Word Handler	$ 129
Magic Window	$ 65
Magic Mailer	$ 45
Magic Words	$ 45
Real Estate Analyzer II	$ 125
Supercalc	$ 185
PFS: Report (New)	$ 65
The General Manager	$ 97
D.B. Master	$ 159
Pascal Programmer	$ 89
Pie Writer	$ 95
Wordstar	$ 219
Datafax	$ 129
Datalink	$ 65
The Home Accountant	$ 52

Peripherals

Micromodem II	$ 269
Novation Apple-Cat II Modem	$ 319
The Joyport	$ 53
The Grappler Plus	$ 129
Game Paddles	$ 28
Ramplus-32K	$ 159
Card Reader	$1219
The Clock	$ 229
Rana Elite One Disk Drive	$ 359
System Saver	$ 65
NEC 3510 Printer	$1699
NEC 12" Green Screen Monitor	$ 169
Microline 84 Printer	$1089
Amdek Color II Monitor	$ 759
Versa Writer Graphics Tablet	$ 239
Ramcard	$ 139
Softcard	$ 269

The Computer Outlet is an associate of The Computer Learning Center For Children. We are experts in educational technology and can customize educational software curriculums for school districts, individual schools, or for the child at home. Please contact us about your software and equipment requirements and feel free to stop by our school in Las Vegas.

We have one of the world's largest educational software inventories featuring our own Computer Learning Center software.

Ten Little Robots (ATARI)	$12.95
Pre-School Math (ATARI)	$19.95

ATARI ™ NEW LOWER PRICES

800 16K	$619
800 48K (with mosaic 32K board)	$714
800 48K (with Atari memory)	$779
400 16K	$269

410 Recorder	$ 75
810 Disk Drive	$439
825 Printer	$579
830 Modem	$155
850 Interface	$169
481 Entertainer	$ 79
482 Educator	$119
483 Programmer	$ 55
484 Communicator	$299
853 16K Ram	$ 79
The Bookkeeper Kit	$169

ATARI Software

CX4104 Mailing List	$ 19
CX404 Word Processor	$115
CXL4007 Music Composer	$ 45
Programming 2 & 3	$ 22
Conversational Languages	$ 45
CX4018 Pilot	$ 59
CX405 Pilot	$ 99
CXL4003 Assembler Editor	$ 45
CX8126 Microsoft Basic	$ 67
CXL4022 Pac-Man	$ 33
CX8130 Caverns of Mars	$ 29
CXL4020 Centipede	$ 33
CXL4006 Super Breakout	$ 28
CXL4008 Space Invaders	$ 28
CXL4009 Computer Chess	$ 28
CXL4011 Star Raiders	$ 35
CXL4012 Missile Command	$ 28
CXL4013 Asteroids	$ 28
The Bookeeper	$105
Home Filing Manager	$ 37
Atari Speed Reading	$ 54
My First Alphabet	$ 26

Business & Utilities

Visicalc	$169
Mail Merge	$ 20
Data Perfect	$ 75
Letter Perfect	$105
Text Wizard	$ 65
Disk Detective	$ 20
Datasm 65 2.0	$ 59
File Manager	$ 65
Syn Assembler	$ 34
Page 6	$ 20
Atari World	$ 39
K-Dos	$ 59
Micropainter	$ 23
Color Print	$ 27
Lisp Interpreter	$ 79
Bishops Square	$ 20
Graphic Master	$ 27
Graphic Generator	$ 17
Basic Compiler	$ 65

Programming Techniques

Display Lists	$ 17
Horiz/Vert Scroll	$ 17
Page Flipping	$ 17
Basics of Animation	$ 17
Player Missile Graphics	$ 24
Sound	$ 17
Data Files	$ 24

For Fast Delivery, send certified or cashier checks, money orders, or direct bank wire transfers. Personal checks allow 2 to 3 weeks to clear. Prices reflect a cash discount only and are subject to change. Shipping—Software ($2.00 Minimum). Hardware—call. Foreign inquiries invited — add 15% for shipping. Nevada residents add sales tax.

TOP SELLERS

Atari

Temple of Apshai	$ 27
Raster Blaster	$ 20
Apple Panic	$ 20
Crossfire	$ 20
Threshold	$ 27
Mousekattack	$ 23
Krazy Shootout	$ 34
Deadline	$ 34
Tumble Bugs	$ 20
Pool 1.5	$ 23
Crypts of Terror	$ 23
Richochet	$ 15
Empire of the Overmind	$ 23
Tanktics	$ 20
Match Racers	$ 20
Wiz & Princess	$ 22
Mission: Asteroid	$ 17
Ali Baba & the Forty Thieves	$ 22
The Shattered Alliance	$ 27
Bug Attack	$ 20
Canyon Climber	$ 20
Shooting Arcade	$ 20
Pacific Coast Highway	$ 20
Clowns & Balloons	$ 20
Ghost Hunter	$ 23
Preppie	$ 20
Rear Guard	$ 17
Lunar Lander	$ 17
War	$ 17
Star Warrior	$ 27
Invasion Orion	$ 17
Dragon's Eye	$ 20
Crush, Crumble & Chomp	$ 20
Jawbreaker	$ 20
Pathfinder	$ 23
Zork I	$ 27
Zork II	$ 27
Action Quest	$ 20
Softporn Adventure	$ 20
Deluxe Invaders	$ 23
Protector	$ 23
Dodge Racer	$ 23
Chicken	$ 23
Nautilus	$ 23
Alien Hell	$ 15
Mar Tesoro	$ 17
Galactic Chase	$ 20
Alien Swarm	$ 23
Intruder	$ 23
Lords of Karma	$ 15
B-1 Nuclear Bomber	$ 12
Rescue at Rigel	$ 20

COMPUTER OUTLET

*** SPECIALS OF THE MONTH ***

ELEPHANT DISKS (BOX)	$ 22
HAYES SMARTMODEM	$229
FRIENDLY TERMINAL SOFTWARE PACKAGE (ATARI)	$ 39
MOSAIC 32K RAM	$ 99
RAMDISK (128K)	$429
AMDEK COLOR I MONITOR	$329
PERCOM DOUBLE DENSITY DRIVE	$679
NEC 8023A PRINTER	$479
K-DOS	$ 59
OS/A +	$ 59
BASIC A +	$ 59
FLIP N' SORT DISKETTE BOX (Holds 50 Diskettes)	$ 21
FLIP-SORT CARTRIDGE BOX (Holds 10 Atari Computer Cartridges)	$ 21
AXIOM GP-100 GRAPHICS PRINTER	$299
AXIOM IMP-4 GRAPHICS PRINTER	$499

Computer Outlet

Park Place — Upper Level
1095 E. Twain — (702) 796-0296
Las Vegas, Nevada 89109

Call Toll Free **800-634-6766**
We accept Major Credit Cards
Mon.-Fri. 8 A.M.-6 P.M.
Sat. 9 A.M.-5 P.M.

NEW Atari

Atari Program Exchange

Salmon Run	$ 19
Galahad and The Holy Grail	$ 19
Seven Card Stud	$ 13
Advanced Music System	$ 23
Elementary Biology	$ 25
Frogmaster	$ 19
Checker King	$ 19

Automated Simulations

Upper Reaches of Apshai	$ 15
Curse of Ra	$ 15
King Arthur's Heir	$ 20
Escape from Vulcan's Isle	$ 20
Crypt of the Undead	$ 20
The Nightmare	$ 20

Broderbund

David's Midnight Magic	$ 23
Track Attack	$ 20
Star Blazer	$ 22
Choplifter	$ 23
Deadly Secrets	$ 23
Stellar Shuttle	$ 20

Datasoft

Spell Wizard	$ 53
Sands of Egypt	$ 27
O'Riley's Mine	$ 23
Rosen's Brigade	$ 23
Fathoms Forty	$ 23

Gebelli

Doctor Goodcode's Cavern	$ 20
Firebird	$ 34
Embargo	$ 34

Innovative Design

Pool 400	$ 27
Speedway Blast	$ 27

JV Software

Ghost Encounters	$ 20

K-Byte

K-razy Kritters	$ 34
K-Star Patrol	$ 34
K-Razy Antiks	$ 34

L & S Computerware

Crossword Magic	$ 34

Lightning Software

Master Type	$ 27

On-Line

Frogger	$ 23
Ulysses and The Golden Fleece	$ 23
Ultima I	$ 27
Ultima II	$ 39

Roklan Corp.

Gorf (d)	$ 27
Wizard of Wor (d)	$ 27
Anti Sub Patrol (d)	$ 20
Gorf (CT)	$ 30
Wizard of Wor (CT)	$ 30
Anti Sub Patrol (C)	$ 15

Sentient

Cyborg	$ 23
Gold Rush	$ 23

Sirius

Space Eggs	$ 20
Sneakers	$ 20
Cyclod	$ 20
Snake Byte	$ 20
Bandits	$ 23
Way Out	$ 27
Fast Eddy	$ 20
Deadly Duck	$ 20
World War I	$ 23
Beanie Bopper	$ 23

Strategic Simulations

Battle of Shiloh	$ 27
Tigers in the Snow	$ 27
The Cosmic Balance	$ 27

Synapse

Slime	$ 23
Shamus	$ 23
Apocalypse	$ 23
Raptillian	$ 23
Picknick Paranoia	$ 23
Claim Jumper	$ 23

such a dictionary is that you can be sure it will pronounce a memorized word correctly.

The Atoms Of English

Much higher quality speech is attainable by using *phonemes*. Phonemes are the "atoms of English," as *S.A.M.*'s manual puts it – the basic sounds upon which all spoken words in the language are based. There are only about 50 or 60 of these.

Both products allow you to define words using special combinations of letters, numbers, and symbols representing these phonemes. For instance, *S.A.M.*'s Reciter has a little trouble pronouncing the word "synthesizer." A much more accurate result can be obtained by leaving the Reciter program and entering the word as a series of phonemes: "SIH4NTHAXSAYZER."

The Voice Box uses a similar set of phonemes. An example for the same word would be "SI2N-TH-ES-UH3-AH2-Y-ZER." Hyphens are used with the Voice Box to separate the phonemes. To add inflection to words and syllables, you use slashes – a forward slash (/) raises the pitch and a backward slash (\) lowers it.

Yes, the phonemes look like alphabet soup, but you must use them for tricky English words if you want accurate speech. Each product lets you vary the pitch, speed, and inflection of speech in enough ways so that virtually any English word is pronounceable. Again, *S.A.M.* does this entirely with software, while the Voice Box has an additional tuning knob which lets you adjust the overall speed and pitch of the speech from slow and guttural to fast and squeaky, very much like changing the speed of a tape recorder.

In addition to pitch control, *S.A.M.* also lets you vary overall speed, and independently stress words or syllables with eight levels of emphasis. Such phoneme-based text is hard to program and read, but it produces some incredibly high-quality speech.

The Voice Box's ten pages of documentation include a phoneme list with example words. *S.A.M.*'s 40-page manual has a very helpful 15-page dictionary of common words and their phoneme equivalents pre-defined for you.

Programs That Talk

Both products allow you to incorporate speech into your own BASIC language programs. You can now have talking aliens, game instructions, audible error messages, and practically anything else you can think of.

Both synthesizers require that their machine language programs be loaded along with your BASIC program and called as subroutines. The text to be spoken is contained in a string variable. Software included with the Voice Box provides a "skeleton" program, complete with the machine language necessary to use the "black box," that you can add to your own program. Alternatively, you could start with the framework program and build your application around it.

S.A.M. "boots" (automatically loads) from a copy-protected diskette. It is simpler to interface with your BASIC program, requiring only one setup statement, and two statements to "call" *S.A.M.* Remember, however, that you must always load the actual *S.A.M.* synthesizer from the special disk. The text-to-speech Reciter program is just as simple to use, but must be accessed from a separate disk you prepare. And since *S.A.M.* is all software, it consumes much more user memory than the Voice Box.

The Atari version of *S.A.M.* blanks out the screen as it speaks, precluding the possibility of synchronizing speech with graphics. However, the original screen image always returns when *S.A.M.* has finished. The Voice Box does not blank the screen, but the software which drives it waits until the speech is done, causing a similar freeze while the box is talking. This can be circumvented with tricky machine language, and documentation is provided to help advanced users access the Voice Box from the machine language level. There also is a way to stop *S.A.M.* from blanking its screen, using a simple POKE, but the result is extremely distorted speech that is impractical for most applications.

Synthetic Shakespeare

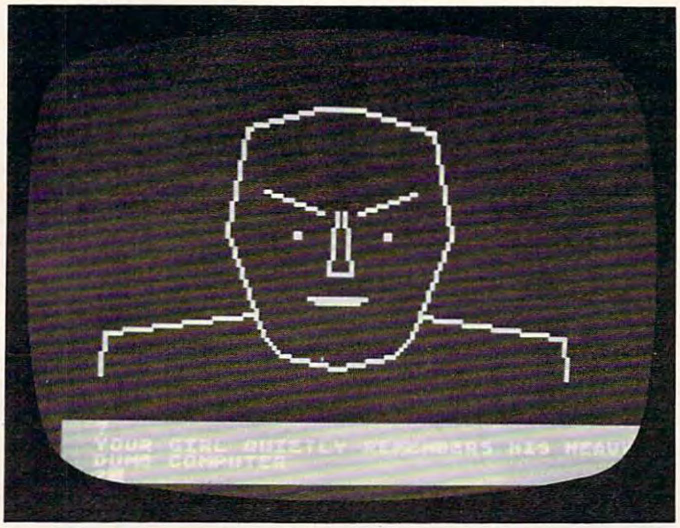
Aside from the machine language driver programs, both products supply various uitlities and demos. *S.A.M.* provides a guess-the-number game, a simple talker program, and a set of four famous speeches – from the Gettysburg Address to Hamlet's soliloquy.

The disk or cassette supplied with the Voice Box includes the aforementioned skeleton program; a "help" demo that shows how to program accurate speech; a "talking head" that lip-syncs with the voice; and two versions of a talker program for 16K or 32K RAM machines. The extended 32K version includes a random sentence generator which utters outrageous phrases, not unlike some of the stream-of-consciousness poetry popular in the 50s and 60s. An example: "That desk quickly loves your rabbit if a ham sandwich sits on my big small girl when your rabbit sleeps."

The Voice Box is (at the moment) the only product usable on cassette-based systems, with abridged support software available on cassette.

A Singing Computer?

Although untested, a singing version of the Voice Box is available for the Apple at a higher price.



The Voice Box's "Talking Face" program babbles in lip-sync with the random sentence generator.

Speech synthesis is remarkable, but a crooning computer is radical. This represents what is sure to be a fascinating future for microcomputers. Who knows? Within a few years, maybe your computer will be reading **COMPUTE!** to you!

Software Automatic Mouth
Don't Ask Computer Software
2265 Westwood Blvd., Suite B-150
Los Angeles, CA 90064
$59.95 Atari (Requires 32K)
$124.95 Apple (Requires 48K)

Voice Box
The Alien Group
27 W. 23rd St.
New York, NY 10010
$169 Atari (Requires 16K or 32K)
$139 Apple (Requires 32K and speaker)
$215 Apple (Includes speaker, ROM dictionary, and singing capability)

©

Review:

VIC-20 Cartridge Games (VIC Firmware)

Harvey B. Herman
Associate Editor

In recent months, a flood of new VIC games has hit the market. Two excellent ones from UMI could keep the kids busy for weeks, or at least until better ones come along.

Spiders Of Mars

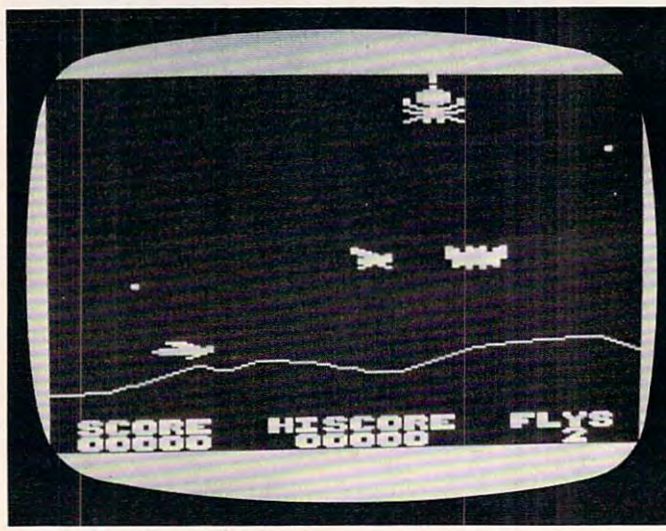
This game is reminiscent of the arcade version of *Defenders*. It begins with a demonstration of the action and allows a choice of skill levels. I like these features in any program. This program also uses color and sound quite effectively, something I look for in all VIC software.

Your character is a fly on the planet Mars. (I normally would not pick a fly as my role model, but this did not detract from the game.) Spiders, hornets, bats, and dragonflies are out to get you, as they would be on Earth. You get three flies (turns) at the start and an extra fly every 10,000 points. The fly is controlled by either the keyboard or a joystick (user's choice).

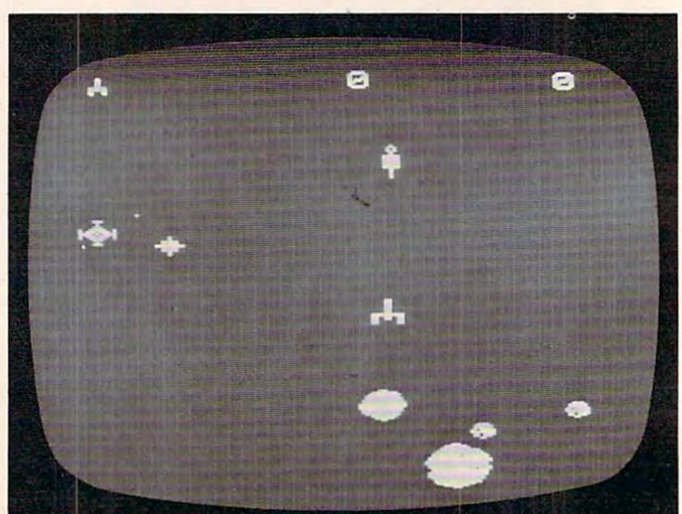
You shoot neutron missiles at the other characters (joystick button or space bar or both), while trying to avoid touching them or being hit by their missiles or smart bombs. Each character hit earns points, and a multiplier is applied at the higher skill levels. When you clear the screen of opponents, the background colors change and the difficulty level increases. Current background colors change and the difficulty level increases. Current score and previous high are displayed continuously. However, during the game the current level of difficulty is not shown.

Let me offer a few hints:

1. My second son believes you can fire faster with the space bar than with the joystick button. He sometimes collaborates with one of his friends, one firing with the bar and the other controlling with the joystick, to rack up some really good scores.
2. Watch out for the bats at the highest skill levels; they get very nasty.
3. Stay away from the top and bottom of the display. Spiders randomly descend from the sky, and fallen ones shoot webs up from the ground.
4. Use the pause button if you develop an acute case of space wrist.



Insect-like aliens on the attack in Spiders Of Mars.



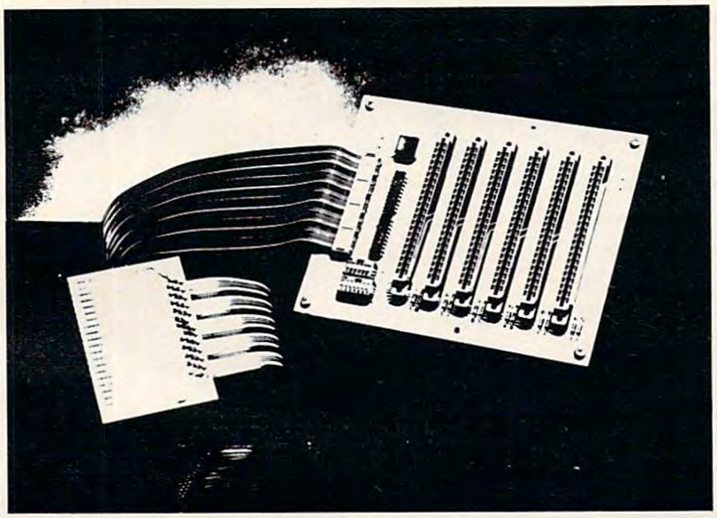
Drifting space rocks and spacecraft in Satellites And Meteorites.

Satellites And Meteorites

This game appears to be modeled after the arcade game *Asteroids*. It begins abruptly without any

An Expansion Interface for the VIC-20

cardboard 6



"CARDBOARD/6"
****

An Expansion Interface
(Motherboard)
For The VIC-20 Personal Computer

The "CARDBOARD/6" is an expansion interface designed to allow the user to access more than one of the plug-in-type memory or utility cartridges now available. Additionally it allows switch selection of games and other programs now available in the cartridge format, without the necessity of turning the computer off and on again, thereby saving a great deal of stress on you, your VIC-20 and your video monitor or television.

Owners of the "CARDBOARD/6" will enjoy these features:

* Accepts up to six cartridges
* Accepts any cartridge in any slot
* Accepts up to 35k of additional RAM memory
* Accepts up to six game cartridges
* Accepts all standard VIC-20 games & utilities
* Allows the user to switch in the desired amount of memory for a particular application
* Allows the user to switch select between up to six game cartridges without turning the computer (saving wear on your VIC, your T.V. and on you)
* Allows the user to switch select various utility programs (ie: programmers' aid, machine monitors and basic utility programs) now available in the cartridge (ROM) format
* Includes a system reset button (the one thing the VIC-20 doesn't have but really needs)
* Allows the capability for future expansion by providing "DAISY CHAINING" capability
* Uses an 18" ribbon cable for hook-up (included) allowing positioning of the unit in a convenient location for easy access to the switches
* Includes a 36 page user's guide which is easy to understand and tells you how to do things that would seem quite impossible but are easy when using the "CARDBOARD/6"

"CARDBOARD/6" is a product of:
CARDCO, Inc.
Wichita, Ks.

"CARDBOARD/3"
****

An Economy Expansion Interface
(Motherboard)
For The VIC-20 Personal Computer

The "CARDBOARD/3" is an expansion interface designed to allow the user to access more than one of the plug-in-type memory or utility cartridges now available. It will accept up to three RAM or ROM cartridges at once. For example some of the possible configurations are :

* 16k RAM + 16k RAM + 3k RAM
* 16k RAM + 8k RAM + Super Expander
* 16k RAM + 8k RAM + Vic-Mon
* 16k RAM + 3k RAM + Programmer's Aid

These and many other usefull memory configurations are available to the VIC-20 user who owns a "CARDBOARD/3".

FEATURES:
=========

We use only high quality T.R.W. gold plated connectors for longer life expectancy and more reliability.

The circuit board is made of a durable aircraft industry standard fiberglass/epoxy compound to prevent accidental damage.

This board is fused to prevent any possible accidental damage to your VIC-20.

This product has a 90 day free replacement warranty covering everything except the fuse.

"CARDBOARD/3" is a product of:

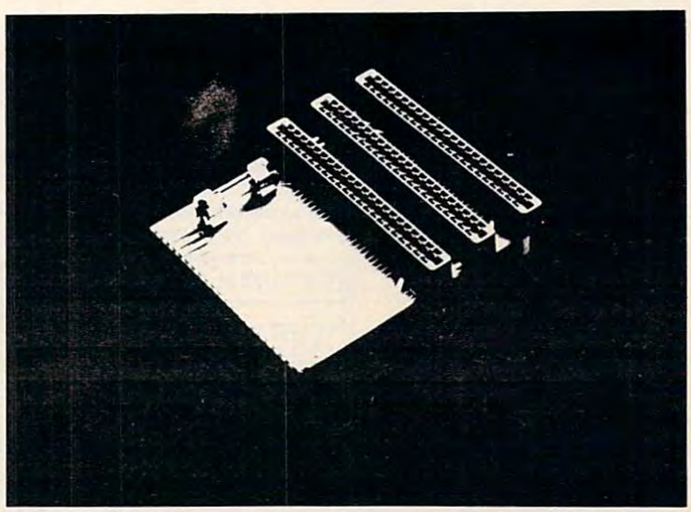
CARDCO, Inc.
Wichita, Ks.

MANUFACTURED IN THE U.S.A.

DEALER INQUIRIES INVITED - PH. (316) 685-9536

An Economy Expansion Interface for the VIC-20

cardboard 3



preliminaries. Your spaceship is being menaced by meteors, satellites (pulsating and twirling ones), and black holes. You shoot and maneuver your ship with a choice of keyboard or joystick. As before, you can see your current score and the previous high. Points are awarded for destroying the attackers, and, if you're good enough, at 10,000 points, a free ship is awarded. (Three are given at the start of the game.)

The game has excellent graphics, but only fair to good sound effects. I was impressed by the explosions of struck meteors into smaller chunks and the 3-D effect as meteors slide by each other. An aggressive satellite has even been known to hide behind an innocuous meteor and spring out at you when the meteor is hit, a nasty surprise.

Two factors make the game difficult to master:

1. The satellites do not move in straight lines.

2. One satellite is shooting randomly, which can cause unexpected hazards (flying chunks).

My testers liked the fact that you are given a new man only after the immediate danger has passed. They felt that the black holes are a unique feature of the game. As you can imagine, it is very difficult to escape from one, but my youngest son claims he did (as yet unverified). They did miss a hyperspace feature which can get you out of some tight spots. Overall, they gave the program a very good rating.

Of the two games, *Spiders of Mars* was the favorite of the kids. An adult would be hard pressed to choose between them. They are excellent games.

Spiders of Mars, *$59.95*
Satellites and Meteorites, *$49.95*
United Microware Industries
3503 Temple Ave.
Pomona, CA 91768

©

Review:

This program can transform any BASIC program into a compiled version that RUNs far faster. For any 4000 or 8000 series models, 4.0 BASIC, except the 8096. There is a version for the new Commodore 64 as well.

Petspeed, An Optimizing Compiler For PET/CBM

Richard Mansfield
Assistant Editor

I have a version of the game "Othello," in BASIC, which is several years old. It's a good opponent, but I avoided playing it very often because it would take so long to figure out its move. After I made a move, a cursor would appear and slowly travel to each square on the board, in an infuriatingly leisurely way. It was like playing with someone who gently put a finger on each square before making a move in checkers.

After it was transformed with *Petspeed*, an optimizing compiler sold by Small Systems engineering, it became a far faster player. Now the cursor flies across the squares in a most computer-like fashion, making up its mind much more quickly than I ever could, as nature intended.

A compiler takes an ordinary BASIC program and creates a second, faster version. The new program is either in machine language or a special machine-language-like code. In either case, the goal is to create a highly efficient program that will RUN far more rapidly.

Petspeed succeeds. Depending on the nature of the program, *Petspeed* can RUN up to 40 times the speed of ordinary PET BASIC. The following simple benchmark took four minutes and one second to RUN in BASIC. The *Petspeed* version took one minute, 33 seconds.

```
10 TI$="000000"
20 FORI=1TO50000
30 X=X+1
40 NEXTI
50 PRINTTI$
```

In operation, *Petspeed* uses a dual disk drive with the target program on a disk (4000 or 8000 series, BASIC 4.0) in Drive One and the special *Petspeed* disk in Drive Zero. It takes over the computer and asks you just one question: what is the filename of your BASIC program? Then, for about 3 1/2 minutes it builds a new version on Drive One in a pseudo-code called "Speedcode" which, when RUN, is used by a *pseudo machine*. In essence, a compiled program is appended to a special "interpreter" program, 8K long, which is loaded into RAM with it. This pseudo machine takes control when you type RUN to use the compiled program.

The compiled program RUNs like a machine language program. If you LIST it, all you see is: "10 SYS (1040) COMPILED IN PETSPEED." The STOP key is disabled (though you can enable it by putting an Enable-Stop instruction in the BASIC program: 10 REM ! ES). You can't use DIM A(N). The N must be a number so the compiler can know in advance how much space to reserve. Since it's no longer BASIC, there is no point to the words RUN or LIST appearing within the program and they, too, are disallowed. These are the only restrictions, however.

Special Options

There are some additional programming techniques not allowed in BASIC. You can use DEF FN with mixed string and numeric arguments. In this way a quick PRINT USING function can be set up. You can declare that *all* characters in a variable name are significant, not just the customary first two. Integer FOR/NEXT loops are permitted. All numeric values are, whenever possible, translated into the faster "integer" type by the compiler anyway.

The "optimizing" feature of this compiler includes the floating point to integer conversion as well as many other improvements. REMs are, of course, dropped, GOTOs and GOSUBs are positioned for maximum efficiency, and all array references are resolved during compilation.

The 32 most frequently used variables in your program are given particular attention. They are set up to be accessed using a rapid addressing mode similar to machine language's zero page addressing. The one most frequently used variable is simply put into zero page.

BASIC programs with machine language patches added to them require special handling. For example, you might need to modify a line which changes the Limit Of Memory pointers to reserve space for machine language. BASIC and

Petspeed use up different amounts of memory. If a machine language subroutine is required, it can be loaded into free RAM space during the program RUN. If the routine involves using BASIC's variables, it will have to be modified to reflect the way that *Petspeed* stores variables. Maps, tables, and descriptions are provided in the *Petspeed* manual to assist machine language programmers with this conversion.

To use the compiler, you must attach a small black plastic box, the "Speedkey," to the First Cassette Drive port on the back of your machine. However, any programs which are compiled into *Petspeed* can run on any machine and do not require the key. If you are interested in selling a program you've compiled with *Petspeed*, you are free to do so. The manufacturer makes no claim on the compiled software and no special keys, boxes, or security devices are necessary.

Petspeed
$350
Small Systems Engineering
222 B View Street
Mountain View, CA 94041
(415)964-8201

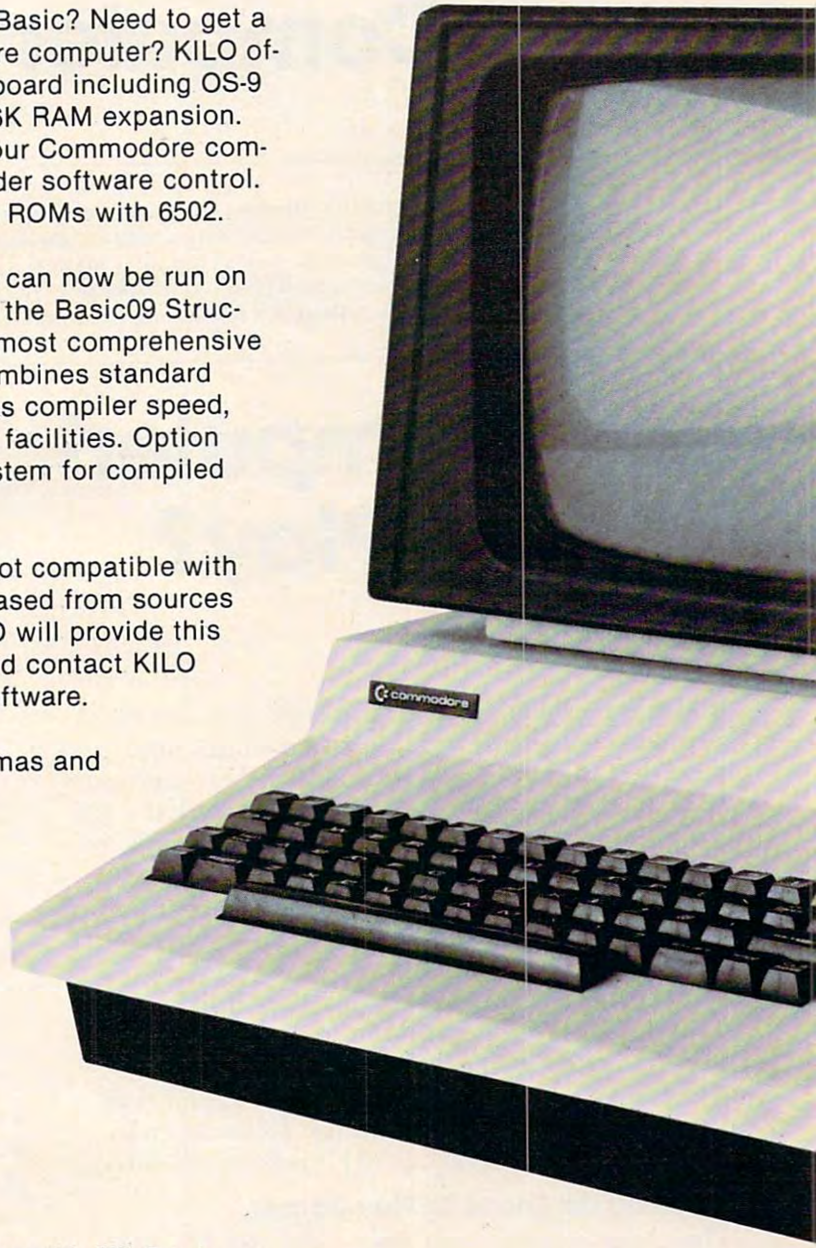
©

SERIOUS CBM® USERS

NEW FROM KILO

"The Serious Solution"

6809 + MICROWARE OS-9* + BASIC09*



Tired of the frustrating limitations of Commodore Basic? Need to get a serious programming language for your Commodore computer? KILO offers The Serious Solution. A 6809 plug in adaptor board including OS-9 Level I Operating System in ROM, Basic09* and 16K RAM expansion. The KILO 6809 adaptor board simply plugs in to your Commodore computer. The switching between 6502 and 6809 is under software control. Expansion RAM can be used to replace protection ROMs with 6502.

With The Serious Solution, all Microware software can now be run on your computer. The standard package comes with the Basic09 Structured Basic Interactive Compiler. The fastest and most comprehensive full Basic language available for the 6809. This combines standard Basic with the best features of PASCAL. It features compiler speed, interpreter friendliness and superlative debugging facilities. Option available includes Run B...a ROMable run-time system for compiled Basic-09.

Please note: The OS-9 distribution disk format is not compatible with the Commodore disk format. OS-9 software purchased from sources other than KILO may need to be reformatted. KILO will provide this service at $10.00 per disk. Software venders should contact KILO for SS-50 to Commodore adaptor hardware and software.

SPECIAL: Buy The Serious Solution before Christmas and get Editor/Assem./Debug package for $200.

The Serious Solution including 6809 adaptor board, OS-9 Level I operating system in ROM, Basic09 and 16K RAM expansion.	$489
Microware* Software Options	
PASCAL compiler	$400
C Compiler	$400
CIS COBOL Compiler	$895
Editor/Assem./Debug package	$300

VISA and Mastercard welcome. Don't forget last month's KILO ad.

CBM and Commodore are registered trademarks of Commodore Computer, OS-9 and Basic09 are registered trademarks of Microware and Motorola. CIS COBOL is a trademark of Micro Focus, Inc.

KILO CORPORATION • P.O. Box 7530 • Ann Arbor, MI 48107 • 313 668-1566

A Monthly Column

The World Inside The Computer

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

Hey, Computer! Wanna Play?

Fred D'Ignazio
Associate Editor

In the September column, I showed you how to develop a computer "friend" program for your child. In the October column, I described computer MAD LIBS® and "dark stories" and included a sample program that encouraged a child to invent his or her own fractured fairy tales. Now, this month, we're going to hook the friend program up to the story-game program – or to any game that will run on your computer.

The entire friend program is included in this column. This new version of the program has been modified and significantly expanded.

The new version of the friend program runs on an Atari 400 or 800 computer. It is written in Atari BASIC and takes up 7217 bytes of memory.

Teaching The Friend To Play Games

The easiest way to teach your computer friend to play games is to make the games part of the friend program.

This seems like a practical solution. You can use empty line numbers 15000 to 32767 for game subroutines. The friend can jump to the games instantly at the child's request.

For short games or for a small number of games, this solution is best. But what happens when you want the friend to play complicated, long games? What happens when you want the friend to know how to play five games, ten games, or more? Then, each time you code a game, you have to code it into the friend. And each time you get a new game, you have to code *it*, too, into the friend. Pretty soon, your program is no longer a friend; it is a blimp.

The solution I have chosen here is to have the friend know the names of up to 50 game programs. If they are stored on disk, the friend calls them and runs them automatically.

At the end of each disk game, you can add a line or two of code that returns control to the friend. When the game is over, the friend automatically wakes up and talks to the child. The friend remembers the child's name and knows he or she has just finished playing a game.

If you have a tape drive (Atari 410 Program Recorder) instead of a disk, you can't easily automate the game-playing process – especially if the child is given the opportunity to select games at random. If each tape contains only one game, the child would have to insert a new tape for each game. If, on the other hand, each tape contained several games, the child would have to wait a long time for a tape indexing program to search through a tape for the chosen game.

Besides, you don't want to completely separate

THE LEARNING CENTER

"Perhaps the finest educational software that I have ever had the pleasure of reviewing. It's easy for kids to use and effective in teaching basic concepts and skills.... My kids are learning with it right now!"

Fred D'Ignazio, Educational Editor—*Compute!*, Associate Editor—*Softside*,
Author of bestseller—*Katie and the Computer*

We hope you've been using your **ATARI** for more than just games ... it is, and can be, a valuable educational tool for you and your children.

Bruce and Dianne Mitchell realized this potential and designed a series of programs for use in their Small World Preschool & Kindergarten located in Durham, N.C. Presented on TV's *PM Magazine* these unique educational programs will introduce your 3-9 year olds to the era of learning with computers. Using the graphics and sound capabilities of the Atari, each program develops a particular skill and reinforces correct responses with happy faces and music.

Beginning with basic concepts such as colors, shapes and the alphabet, your child will progress to an understanding of counting, arithmetic, and language skills.

Widely acclaimed, classroom designed and tested, these unique educational tools are now available to you ... *for your children.*

SPECIAL SKILLS	**Preschool**	**K**	**1**	**2**
* Color For The Non-Reader	●	●		
* Name That Color		●	●	
* Like Shape Identification	●	●	●	
* Different Shape Identification	●	●	●	
Cave Game	●	●	●	●
MATH AND NUMBER SKILLS				
Count With Me	●	●	●	
Number Recognition	●	●	●	●
Addition		●	●	
Subtraction		●	●	
Add.—Vertical/Horizontal		●	●	
Sub.—Vertical/Horizontal		●	●	
Advanced Addition/Subtraction		●	●	●
Ones and Tens			●	●
LANGUAGE SKILLS				
* Alphabet Recognition			●	●
* Letter Sequence	●	●	●	
* Like Symbol Discrimination	●	●		
* Different Symbol Discrimination	●	●		

Pricing Information	Cass.	Disk
Single Program	6.95	9.95
Special Skills	24.95	29.95
Language Skills	19.95	24.95
Math and Number Skills	34.95	39.95
Complete Set	59.95	64.95
Edumate™ Light Pen	19.95	

	Cass.	Disk
Pre-School	36.95	41.95
Kindergarten	54.95	59.95
First Grade	49.95	54.95
Second Grade	21.95	26.95

Add $2.00 for postage & handling

* Compatible with our Edumate™ Light Pen
NOTE: All software requires 8K cassette/16K disk

Start competing with your 4-year-old for computer time. Enroll in THE LEARNING CENTER, it could be the best investment you ever make ... *in your children's future.* See your local dealer or order direct.

Available now for the ATARI 400/800
COMING SOON for the TRS-80 Model I, III, Color Apple, VIC, TI-99
Free Catalog Upon Request

THE PROGRAMMER'S INSTITUTE
—a division of **FUTUREHOUSE™**
P.O. Box 3191 Dept C
Chapel Hill, NC 27514

1-919-489-2198 10 am - 9 pm, Monday - Saturday your kids from the computer. Even preschoolers can be taught the basics of operating a tape drive. After all, they need to know only three commands: CSAVE, CLOAD, and RUN. And they need to know only five buttons: REWIND, PLAY, STOP/EJECT, and RECORD on the recorder, and RETURN on the computer. I believe it is better to teach the kids how to run the recorder and the computer, and let them load the game tapes themselves.

I *do* recommend one thing, though: buy ten-minute tapes (five minutes to a side) and store only two games per tape – one per side. You can keep the tapes in cases inside the kid's Buster Brown shoe box. Each tape case should have a bright, colorful label on the top. For toddlers, you might use geometric shapes instead of words as labels.

Why use such short tapes? I once asked the same question. That was before I tried to store 25 game programs on a 60-minute tape. Finding and using all the games was a nightmare, even with up-to-date pointers to the game locations. And it took forever. If you try this, your child might, at the beginning of a tape search, be on the seat waiting eagerly for a game, but he won't be at the end. He'll be gone.

Kids don't have the patience to locate and load programs stored on long-playing tapes. And neither should you. Besides, positioning the tape using the tape counter, the REWIND button, the ADVANCE button, and the STOP/EJECT button is hard! It can be a supremely frustrating experience for little kids.

So loosen your purse strings and buy a bag of short tapes. After just a few games, you and your kids will be glad you did.

One more thing. The friend program could have been more automated, even using tapes. On the Atari computer, you can save your game programs onto tape using the SAVE "C:" command. And you can run programs stored this way by typing RUN "C:". These commands can be built right into the friend program and at the tail end of all the game programs.

This method would save children from having to type CLOAD and RUN. They would just have to load a tape into the recorder, rewind the tape to the beginning, and press the PLAY button. The computer would do the rest.

On the other hand, using this method, the programs load much slower than the normal method. (Sometimes it takes them almost twice as long.)

I think speed is critical for a young child. Wherever possible, the child should not be kept waiting while the computer goes about its chores. Thus, in the friend program in this article, I chose the quicker normal method (CSAVE/CLOAD).

A Face Lift For An Old Friend

This section is for those of you who have already loaded the September version of the friend into your Atari computer. There are many changes to the old program, but there is no point in entering in the entire program a second time.

On the other hand, if you haven't loaded the old version of the friend program into your computer, that's fine. The entire program is listed at the end of this column. Just type it in.

Program Documentation (lines 10-95): REM comment lines introduce the friend and describe its major functions.

Dimension Variables (lines 100-130): Delete old line 120. Add new lines 120, 125.

Friend Master (lines 500-600): It is probably better just to retype these few lines. Almost all are changed in some way.

Pay special attention to the new line 510. This line checks to see if the friend has already been called on. If so, the friend locates the child's name in RAM and skips the normal wake-up routine.

The new line 550 calls the game-playing subroutines.

Friend Wake-Up/Talking (lines 1000-3110): This section is almost identical to the old version of the friend program. It causes the friend to wake up and gives the friend the ability to talk using DATA statements stored at the tail end of the program.

Move old line 1010 to 2012.
Move old line 2006 to 2011.
Delete old line 2005.
Add new line 2010.

New lines 2010 and 2011 turn this subroutine into a general-purpose "friend-talker." The friend can now get "talk" messages (DATA statements) at any location.

Subroutines that call the "friend-talk" subroutines point them to the right location. For example, the game subroutine points the friend to lines 12000-12999 to talk about games. The wake-up subroutine points the friend to lines 10000-10999 to give its wake-up greetings. If the friend is already awake, another subroutine points the friend to lines 11000-11999 to greet the child after a game. The tape-load subroutine points the friend to lines 13000-13999 so the friend can tell the child how to load a game tape.

Whenever you add new messages for the friend to tell the child, just add the messages at an unoccupied set of line numbers (anywhere from 14000 on up), and remember to call the friend-talk subroutine at line 2010. But before you call the subroutine, set the individual-message pointer, N, to 1; and set the type-of-message pointer, DAT-

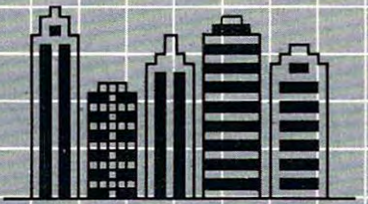
TODAY'S MENU

For Your VIC™ 20 and ATARI® 400/800

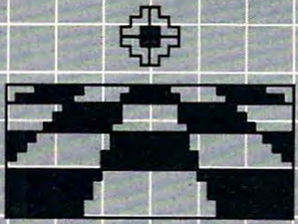
Games

ASTROBLITZ
Protect your planet by destroying enemy saucers.

TRASHMAN
Drive the garbage truck and empty the city's trash cans. But watch out for the flies.



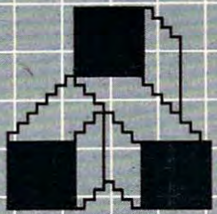
CITY BOMBER
Level a city to make it easy to land. Take off and do it again.



ACTION GAMES
SEAWOLF, BOUNCEOUT, or VIC TRAP. You'll need sharp eyes and quick hands for these.

Education

EDUCATIONAL/RECREATIONAL (I & II)
Put fun into learning math, spelling, and spatial relationships.



LOGIC GAMES
The computer is thinking. You should be, too.

Choose an Entree:

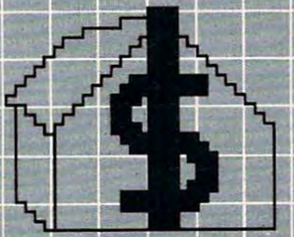
Maybe it's a colorful and challenging game like ASTROBLITZ, TRASHMAN, or CITY BOMBER; perhaps an educational game like HANGMAN or MATH HURDLER; maybe a basic diet of household concerns like HOME INVENTORY, HOUSEHOLD FINANCE, or DECISION MAKER.

Take your pick. All you have to supply is your VIC™ 20 or ATARI® 400/800 and your own ideas about how to put our software to use. We supply the rest: competitive games, educational games, down-to-earth personal programs, and simple instructions that make everything easy to digest.

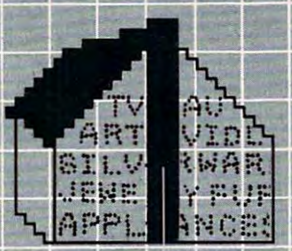
What you see here is part of the menu for today. We'll be adding more, so check our menu from time to time. It's growing, and it's all take home.

Contact your local outlet.

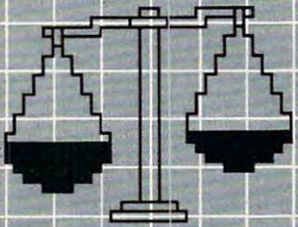
Personal



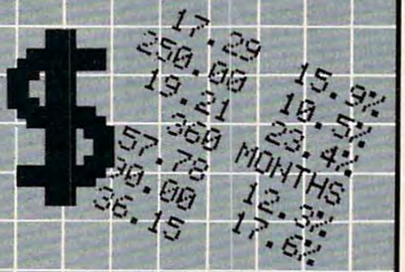
HOUSEHOLD FINANCE
Schedule the family budget, account for expenditures, and face the tax man with a smile.



HOME INVENTORY
Make a record of what you own and revise it as you go. Find out what you've got in seconds.



DECISION MAKER
What should you do about virtually anything? Here's help.



LOAN ANALYZER
Home, car, boat . . . whatever. How much can you afford? Find out.

CREATIVE SOFTWARE
A Division of ASCI, Inc.
201 San Antonio Circle, Mountain View, CA 94040 U.S.A. (415) 948-9595

NUM, to the line number where you've added new messages (e.g., DATNUM=14000).

On line number 2033, I changed the delay loop from 800 to 200. This adjusts the amount of time a single friend message will remain on the TV screen. You can decide how long the messages should stay, based on the reading level of your child.

Friend Asks a Question (lines 3200-3620): Several lines have changed, so it's easiest to delete all old lines, from 3200-3390. Then type in all new lines, from 3200-3620. These new lines convert the subroutines into general-purpose question-*askers*.

Also, they add a significant improvement to the old friend program. When children type in an answer – say their name – and make a mistake typing, they can now type the DELETE key, erase the erroneous characters, and type their answer correctly.

Wake-Up Bell/Friend Voice (lines 4000-4880): These subroutines are almost identical to the old version of the friend program.

Delete old lines 4625 and 4770. These lines are fossils from a much older friend program that is now extinct.

Friend's Face (lines 5000-5550): Identical to the old version of the friend program. (So no face lift is required, after all!)

These subroutines draw the friend's face. They animate its eyes for sleeping, winking, and waking. They animate its mouth for talking.

Lines 6000 and Up: This part of the program is completely new, with one exception: the old DATA statements, lines 6000-6022. Renumber statements 6010-6022 to 10010-10022 by changing the "6's" to "10's".

Next change line 6000 to 10005.

Add new line 10000.

Remember to add new lines 10030-10032.

Friend's Games (lines 6000-7070): On lines 6000-6220 the friend asks the child to play a game. If the child wants to play, the friend displays the list of up to 50 games you have placed (at ten-line intervals) on lines 12030 to 12520.

The friend displays the name of each game and waits to see if the child wants to play the game. If not, the friend goes on to the next game. If the child doesn't like any of the games, the friend prints an "I'm sorry" message and says good-bye.

If the child *does* want one of the games, the friend goes to the subroutine at 6310 and stores the child's name in a secure spot in the computer's memory. Then it goes to the subroutine at 6410 to see if the child needs help loading the game tape. If the child needs help, the friend explains the steps the child must follow to load a tape. The friend goes over the instructions until the child has them all straight. Then the friend says good-bye.

After playing a game, the child reloads and runs the friend program. On line 510, the friend automatically checks to see if it has spoken to the child earlier. If so, the friend jumps to the subroutine at line 7010 and retrieves the child's name. The friend appears on the screen already awake and greets the child by name. Then it goes back to the friend master to see if the child wants to play another game (see line 550), or to do some future activity (empty lines 560-999) that you can add later on.

Friend Messages (lines 10000-10341): The wake-up messages are on lines 10000-10032.

The friend's greeting messages after the child returns from a game are on lines 11000-11015.

The friend's list of games and its "I'm sorry" message are on lines 12000-12536.

The friend's instructions for loading a tape are on lines 13000-13041.

The messages are structured as in the old friend program. A new message begins on every tenth line number. The message begins with a DATA statement and a single number (like 1, 2, 5, or 6). The number indicates the number of screens (see SNUM on line 2012) in the current message.

Following the DATA statement with the number of screens are the DATA statements which contain the message text. Each DATA statement contains one screen of messages: from one to four message lines, each line containing a maximum of nine characters. Each DATA statement (screen) ends with a marker – a minus one (-1). The child's name can be included in the message by placing the asterisk token (*) at the appropriate place. (For example, see lines 10022 and 12531.)

When you are building messages, a good rule of thumb is to display only one word on each line. This makes it easier for the child to read the message, and makes the screen look simpler and less cluttered. You do this by following each word with a comma. For example, the command DATA I, HOPE,YOU,-1 will cause the friend to display one screen with only one word per line.

Returning From A Fairy Tale

Last month's column included a "fractured fairy tale" program for you and your kids to try. The program ran independently of the computer friend. Now I'm going to show you how to modify the program so it can point the child back to the computer friend.

First, delete the old lines beginning with line 1435. Then add the following lines:

```
1435 GRAPHICS 0
1436 REM ***
1437 REM *** RETURN TO FRIEND
1438 REM *** PROGRAM--ON TAPE
```

Recreational Computing Back Issues

Recreational Computing was the first and only personal computing magazine when it started in 1972 (it was called the *PCC Newspaper* back then). Bob Albrecht, David Thornburg, Isaac Asimov, Don Inman, Ramon Zamora, Robert Jastrow, Mac Oglesby, Adam Osborne – the list of authors reads like a Who's Who of microcomputing. These and many other authors contributed some of the finest articles about computers and now-classic games to the pages of *Recreational Computing*.

Last fall, *Recreational Computing* was merged into **COMPUTE!** and we are now offering available back issues. Whatever your interest, you'll find something here – from Spanish BASIC to Computers in Sports Medicine, from Future Fantasy Games to Robot Pets.

September 1974 A Practical, Low-cost Home/School Microprocessor System, The Computer Illiteracy Problem, Eight Games In BASIC

March 1975 Build Your Own BASIC, The Computer In Art, Biorhythms

March/April 1976 A TTY Game, Games With The Pocket Calculator, Dodgem, Square, Tiny BASIC To Go

July 1976 BASIC Music, Tiny Trek For Altair, 16 Bit Computer Kit, Musical Numbers Guessing Game, Programmer's Toolbox

September/October 1976 Computer Games In The Classroom, Planets Game, Dungeons And Dragons, Hats Game, Pythagoras And Rational Music

November/December 1976 Story, Snake, Packl, Frogs Games, Make Believe Computers, The First West Coast Computer Faire, Subroutines, The First Computer

January/February 1977 Robot Pets, Computers And Space, Tiny Languages, Teaching Using Conversational Programming, High School Computers, Reverse, Tiny PILOT, Mastermind

March/April 1977 Z-80 PILOT, 6502 Assembly Programming, Tiny BASIC For Beginners, Math Drills & Games, Community Information Systems, Mine, Sales Simulation, Native American Board Games

July/August 1977 Do-it-yourself CAI, Pet Robots: New Capabilities, PILOT, CAI In BASIC, Programming The HP-25, Capture, Inverse Reverse, 8080 Matrix Subroutines, Women And Computers

September/October 1977 The $595 PET, More Tiny Languages, Computer Networks, The Bead Game, Biofeedback And Microcomputers Part 1, Home Energy Management, Sandpile Game, A BASIC PILOT

January/February 1978 Pascal Vs. BASIC, COMAL: Structured BASIC, Video Disks: Magic Lamps for Educators?, A Computer Revolution?, Pounce, The Mechanics of Robots, TRS-80: A Status Report

March/April 1978 Epic Computer Games, Micros for the Handicapped, Buckets Game, Prayer Wheel Program, Computer Contagion, Measuring Time, Frog Race, The IBM 370 Model 69

July/August 1978 Computer Whiz Kids, Public Access To Computers, Man-made Minds, Post-human Intelligence, A Modern-day Medicine Show, Live Wire Design, ASCII Graphix, Baseball, Concentration, Gambler's Paradox

September/October 1978 Kingdom Game, Computers and Museums, Sorcerer of Exidy, Snooping With Your PET, APL, Decimals in Tiny BASIC, Apple Math, TRS-80 Level II, A Grown-up Field Evaluation

November/December 1978 APL Games, The Return of the Dragons, Animated Games for TRS-80, Runequest, All In The Mind, The L-5 Society, Phantnum, Some Guidelines for Microcomputer Chess, Dataman

January/February 1979 A Jules Verne Fantasy, Artificial Intelligence, The Apple Corps is With Us, TRS-80 Personal Software, Vending Machine Gets "Brain," Apple II I/O, The Memory Game, REINO: Spanish Kingdom

March/April 1979 Calculator Comics, "Lord of the Rings," Chess Reconsidered, Database, Beastiary, Color Your Own Graphics, Universe, Easy POKEing with Applesoft BASIC, Air Raid, TRS-80 3-D Plots, Slot, Apple Rose

May/June 1979 PILOT for Apple II, The Game of Life, Gold Handicapping, Hunt, BASIC vs. Pascal, Inspector Clew-so, Flash for SOL, Faster Jumble, Concept Sans Computer, A Beginner's Guide To FRP

July/August 1979 Summer Fun, Fooling Around With Your PET, Cryptarithms, Baseball, Newett Awl's Goat, Zork: A Computerized Fantasy Simulation Game, What Light on Yonder Panel Flashes, The Dedicated Word Processor, The FORTE Music Programming Language

September/October 1979 TRS-80: Outside Connection, The Architecture of Multi-Player Games, The Sounds of Texas Instruments, Dynamic Color Graphics on the New Atari, An Apple PILOT, Gandalf, Spanish BASIC, Designing Animal Games, APL Mastermind

November/December 1979 SHOGI: Games For You To Program, Atari Sounds, Texas Instrument Graphics and Animation, Interrupt, Match Me, Calendar, Making Music on the PET, Tower of Hanoi, Bingo, Animal Games

January/February 1980 Computing and Holistic Health, TI Graphics and Animation Part 2, Games To Program, New Directions in Numerical Computing, An Extended BASIC "IF" Facility, Beating Computer Anxiety, Capture for PET, 8080 Tic Tac Toe, Chainwalk, Programming Problems

March/April 1980 Special Games Issue: Recreation Apple II Hi-res Graphics, Delicious Functions, Galaxy II, Fairy Chess, Raging Robots, Program Instruction Builder, Data Retrieval: An Introduction

May/June 1980 Introduction to Computer Music, CBBS Phone Numbers, 6502 Machine Language, The Electric Phone Book, Number Translation, Sea Search, Apple Animation, Twister Move Generator, DOZO, Shell Game, Home Video Displays, A Proposed Graphics Language

July/August 1980 Fantasy Games Issue: Write Your Own Computer Fantasy Simulation, Wizard's Castle, On Future Fantasy Games, Wonderful World of Eamon, In Defense of Hackers, Touch Panels and Interactive Graphics

September/October 1980 Probability Trees: Big Business on the Micro, The Best of People's Computer Company, Computer Analysis of Athletics, Word Search, Computers in Sports Medicine, Wired, Revolution in Typography?, Textrapolation

November/December 1980 Computerized Voting, Computer-Using Educators, Hot-rod Computers, House of the Future, Yote, DOZO in Pascal, What is Truth?, Sixth Order Magic Squares on a TRS-80

January/February 1981 The Education Revolution: Computer Games in the Classroom, An Art-producing Turtle, Computer Literacy Resources, Musical Compositions Using Computers, Microcomputers in China, Twenty Questions, The Pirate's Life for Me, Computers and the Volcanic Fallout

March/April 1981 Space Exploration: Frontiers for You and Your Micro, Voyage To Antares, A Spaceship Simulator, The Computer as Chess Ally, Star Trek – A Dialogue Approach, Mark of Breeding (fiction), The Fifteen Puzzle

May/June 1981 Using Computers at Sesame Place, Atari PILOT and Turtle Graphics, Computer Anatomy for Beginners, The Impact of Micros, Nevada-style 8-spot Keno, Sketch Pad, Sum of the Digits, TRS-80 Property Management Program, The Pocket Corner

July/August 1981 Which Computer Should You Buy?, Commodore's New Rainbow Machine, The Wired Nation: Do We Want It?, Computers at the Junior Museum, 3-D Tic Tac Toe for PET, Number Crossword for all Computers

September/October 1981 43 Ways To Make Money With Your Micro, How To Start A Software Exchange, Who Are Computer Criminals?, Micros Behind Bars, Number Systems, Computer Knock-knock Jokes in BASIC and LISP, For Photographers Only, Fibonacci Nim, Roman Numeral Conversum Programum

Special *Recreational Computing* Back Issue Pricing:

Single Issue:	**$ 3.00**	**Any Fifteen Issues:**	**$25.00**
Any Five Issues:	**$10.00**	**Any Twenty Issues:**	**$30.00**
Any Ten Issues:	**$20.00**		

For Fastest Service, Call Toll Free **800-334-0868**
In NC Call **919-275-9809**.

Or Send Order and Payment to **COMPUTE!** Publications, P.O. Box 5406, Greensboro, NC 27403.

In the US, please include $.20 per issue ordered for shipping and handling. Outside the US, please include $.30 per issue for surface mail. Orders must be prepaid in US funds or international money order. All orders subject to availability.

```
1439 REM ***
1440 PRINT "TIME TO WAKE UP YOUR FRIE
     ND!":PRINT
1442 PRINT "DO YOU NEED INSTRUCTIONS"
     ;:INPUT ANSWER$
1443 IF ANSWER$="N" THEN GRAPHICS 0:G
     OTO 1530
1445 IF ANSWER$<>"Y" THEN 1435
1446 GRAPHICS 0
1450 PRINT "PUT 'FRIEND' TAPE INTO"
1460 PRINT "PROGRAM RECORDER."
1465 PRINT
1470 PRINT "REWIND TAPE TO BEGINNING.
     "
1475 PRINT
1490 PRINT "TYPE 'CLOAD' AND PRESS RE
     TURN."
1495 PRINT
1496 PRINT "PRESS PLAY BUTTON"
1497 PRINT "ON THE PROGRAM RECORDER."
1498 PRINT
1499 PRINT "PRESS RETURN AGAIN."
1500 PRINT
1505 PRINT "WHEN COMPUTER IS FINISHED
     "
1506 PRINT "LOADING THE 'FRIEND' TAPE
     ,"
1508 PRINT "THE COMPUTER WILL TYPE 'R
     EADY'."
1509 PRINT
1510 PRINT "THEN YOU TYPE 'RUN'"
1515 PRINT "AND PRESS RETURN."
1520 PRINT
1530 END
```

The above commands help the child exit from the fairy-tale program and reload the computer friend – *if the friend and the fairy tale are stored on tape*.

On the other hand, if you have a disk drive, you can make all transfers to and from the friend automatic. The friend can start games automatically. And the games can automatically reload and run the friend.

To modify the *friend* program is simple. First, you change line 90 to read:

90 REM * DISK VERSION OF FRIEND**

Second, you change line 6180 to read:

6180 IF M$(1,1)="Y" THEN GOSUB 6310:GOTO 6410

Third, you delete old lines 6400-6470 and add the following new lines:

```
6350 RETURN
6400 REM *** DISK VERSION OF FRIEND
6405 REM ***
6408 REM *** SELECT GAME PROGRAM/EXIT
       FRIEND
6410 GOTO 6410+Z*10
6420 RUN "D:TELLTALE"
```

The only game program currently referenced is "Telltale," at line 6420. Telltale is the name I have given the fairy-tale program (from last month's column) that is stored on disk.

You can have the friend automatically run up to 50 game programs by adding their full (English) names to the friend's game list on lines 12030-12520. You add each new game after an interval of ten lines (12040, 12050, 12060, etc.). The format you follow is the same as in the fairy-tale game, Telltale, listed at lines 12030 and 12031:

12030 DATA 1
12031 DATA STORY,GAME?,-1

At 12030, the Data statement tells the message-display subroutine at line 2010 that there is only one screen in this message. At 12031 is the English name of the game as it will be displayed on the screen by the computer friend. The game name is followed by a question mark since the friend is asking if the child wants to play this particular game.

Next, so the friend can actually load and run the new game, you need to add the game's *program name* to lines 6430 and up. You separate each program name by ten lines (6430, 6440, 6450, etc.). You follow the same format as the fairy-tale program, Telltale, on line 6420:

6420 RUN "D:TELLTALE"

Remember, you can add up to 50 game programs for the friend to automatically run.

Calling On A Friend

Now you know how the computer friend automatically loads and runs a game. But how does a game reload and run the friend?

You can learn how by changing the fairy-tale program, Telltale. First, delete old lines 1460 and 1470. Second, add the following new lines:

```
1480 REM ***
1490 REM *** RETURN TO FRIEND
1500 REM *** PROGRAM--ON DISK
1510 REM ***
1520 RUN "D:FRIEND"
```

As you can see, it's simple. The only real command you add is RUN "D:FRIEND".

By following the instructions above, you can add dozens of games to your computer friend's repertoire. The friend runs the games automatically, and the games automatically return control to the friend when your child is through playing them.

Acknowledgments And Predictions

I admit I haven't been too good about predicting where this column is going each month. Like a red-faced weather forecaster, I apologize for the times my predictions haven't come true.

Unlike the weather forecasters, I am going to stop making long-range forecasts. I'm going to stick to the near-term – namely, next month. Next month I plan to show you how to teach the computer friend how to remember things. After you modify the program, the friend will remember facts about itself (its name, shape, hair color, favorite jokes, etc.) and facts about the child. When the child tells the friend something important, the friend will remember.

I'd like to thank Bruce Mitchell for his valuable programming assistance. Bruce is the chief "big-person" programmer at the Small World Preschool and Kindergarten in Durham, North Carolina.

MAD LIBS® is a registered trademark of Price/Stern/Sloan, Publishers, Inc., Los Angeles, CA.

```
10 REM ***************
20 REM COMPUTER FRIEND
30 REM ***************
40 REM
50 REM *** WHEN PROGRAM BEGINS,
55 REM *** THE FRIEND'S FACE
60 REM *** APPEARS.  THE FRIEND
65 REM *** WAKES UP AND GREETS
70 REM *** THE CHILD.  THE FRIEND
75 REM *** THINKS UP COMPUTER GAMES
80 REM *** THE CHILD CAN PLAY.
85 REM
90 REM *** TAPE VERSION OF FRIEND
95 REM
100 REM *** DIMENSION VARIABLES
110 DIM M$(9):REM * MESSAGE
120 N=1:REM * INDIVIDUAL-MESSAGE POIN
    TER
125 DATNUM=10000:REM * TYPE-OF-MESSAG
    E POINTER/10000 = FRIEND WAKE-UP
130 DIM NAME$(9):REM * CHILD'S NAME
500 REM *** FRIEND MASTER
505 GRAPHICS 2+16
510 IF PEEK(1791)=1 THEN GOSUB 7010:G
    OTO 550
515 GOSUB 1010:REM * FRIEND WAKE-UP
520 GOSUB 2010:REM * FRIEND INTRODUCE
    S HIMSELF/HERSELF
530 ANSWER=2:GOSUB 3210:REM * FRIEND
    LEARNS CHILD'S NAME
540 GOSUB 2010:REM * FRIEND HAPPY TO
    SEE CHILD
550 GOSUB 6010:REM * PLAY GAME?
600 END
1000 REM *** FRIEND WAKE-UP
1010 GOSUB 5010:REM * DRAW FACE
1020 GOSUB 5410:REM * DRAW SLEEP EYES
1030 GOSUB 5210:REM * DRAW CLOSED MOU
     TH
1035 FOR P=1 TO 800:NEXT P
1040 GOSUB 4010:REM * WAKE-UP BELL
1050 GOSUB 5460:REM * DRAW OPEN EYES
1060 FOR P=1 TO 600:NEXT P
1070 GOSUB 5320:REM * WINK EYE
1080 FOR P=1 TO 100:NEXT P
1085 M=0:GOSUB 4820:REM * WINK NOISE
1090 GOSUB 5460:REM * DRAW OPEN EYES
1100 FOR P=1 TO 800:NEXT P
1120 RETURN
2000 REM *** FRIEND TALK
2010 RESTORE DATNUM+N*10:REM * SELECT
     MESSAGE
2011 N=N+1:REM * SET POINTER TO NEXT
     SET OF FRIEND MESSAGES
2012 READ SNUM:REM * SNUM = NUMBER OF
     SCREENS IN CURRENT SET OF FRIEN
     D MESSAGES
2015 FOR K=1 TO SNUM
2020 GOSUB 3010:REM * FRIEND TALK--1
     SCREEN
2033 FOR P=1 TO 200:NEXT P
2035 GOSUB 5510:REM * CLEAR MESSAGE W
     INDOW
2040 NEXT K
2050 RETURN
3000 REM *** FRIEND TALKING--1 SCREEN
3010 PY=2:REM * MESSAGE VERTICAL (Y)
     START LOCATION
3020 PY=2:REM * MESSAGE VERTICAL (Y)
     START LOCATION
3030 PX=14:REM * HORIZONTAL (X) CENTE
     R OF MESSAGE ON SCREEN
3040 READ M$
3050 IF M$="-1" THEN RETURN
3051 IF M$="*" THEN M$=NAME$
3055 GOSUB 5260:REM * OPEN MOUTH
3060 POSITION INT(PX-(LEN(M$)/2)+0.5)
     ,PY:REM * CENTER LINE
3070 PRINT #6;M$
3075 GOSUB 4810:REM * FRIEND SOUND
3080 FOR P=1 TO 10:NEXT P:REM * KEEP
     MOUTH OPEN
3090 GOSUB 5210:REM * CLOSE MOUTH
3095 FOR P=1 TO 50:NEXT P:REM * KEEP
     MOUTH CLOSED
3100 PY=PY+2
3110 GOTO 3040
3200 REM *** FRIEND ASKS CHILD A QUES
     TION
3210 OPEN #1,4,0,"K:"
3212 M$=""
3215 POSITION 11,4
3217 FOR I=1 TO 9
3220 GET #1,A
3222 IF A=126 AND I=1 THEN 3220
3225 IF A=126 THEN GOSUB 3310
3230 IF A=155 THEN 3265
3240 PRINT #6;CHR$(A);
3250 M$(LEN(M$)+1)=CHR$(A)
3260 NEXT I
3265 FOR P=1 TO 75:NEXT P
3267 GOSUB 5510:REM * CLEAR MESSAGE W
     INDOW
3270 CLOSE #1
3280 GOSUB 3410:REM * EVALUATE ANSWER
3290 RETURN
3310 POSITION I+9,4:PRINT #6;" ";
3312 POSITION I+9,4
3315 M$(LEN(M$))=""
3317 I=I-1
3320 GET #1,A
3330 IF A<>126 THEN 3390
3350 IF I<2 THEN 3320
3360 GOTO 3310
3390 RETURN
3400 REM *** EVALUATE ANSWER
3410 ON ANSWER GOSUB 3510,3610
3420 RETURN
3500 REM *** NO NEED TO STORE ANSWER
3510 RETURN
3600 REM *** ANSWER=CHILD'S NAME
3610 NAME$=M$
3620 RETURN
4000 REM *** WAKE-UP BELL
4010 BEL=105:TIM=7.5:GOSUB 4040
4020 BEL=132:TIM=8.5:GOSUB 4040
4030 SOUND 0,0,0,0:RETURN
4040 VLM=15:INC=0.79+TIM/50
4050 SOUND 0,BEL,10,VLM
4060 VLM=VLM*INC
4070 IF VLM>1 THEN 4050
4080 RETURN
4800 REM *** FRIEND VOICE
4810 M=INT(RND(1)*51)+15
4820 FOR A=M+25 TO M STEP -8
```

```
4830 SOUND 0,A,10,10
4840 FOR T=1 TO 10
4850 NEXT T
4860 NEXT A
4875 SOUND 0,0,0,0
4880 RETURN
5000 REM *** FRIEND'S FACE
5010 GRAPHICS 2+16
5040 POSITION 2,1:PRINT #6;"
     {3 SPACES}*"
5050 POSITION 2,2:PRINT #6;"  / \"
5060 POSITION 2,3:PRINT #6;" ====="
5070 POSITION 2,4:PRINT #6;"/
     {5 SPACES}\"
5090 POSITION 1,6:PRINT #6;"<:  ^  :>
     "
5100 POSITION 2,9:PRINT #6;"\_____/"
5110 RETURN
5200 REM *** CLOSE MOUTH
5210 POSITION 2,7:PRINT #6;":
     {5 SPACES}:"
5220 POSITION 2,8:PRINT #6;": --- :"
5230 RETURN
5250 REM *** OPEN MOUTH
5260 POSITION 2,7:PRINT #6;": ___ :"
5270 POSITION 2,8:PRINT #6;": \_/ :"
5280 RETURN
5300 REM *** LEFT EYE WINK
5320 POSITION 2,5:PRINT #6;": O - :"
5330 FOR P=1 TO 150:NEXT P
5340 RETURN
5400 REM *** EYES ASLEEP
5410 POSITION 2,5:PRINT #6;": - - :"
5440 RETURN
5450 REM *** EYES AWAKE
5460 POSITION 2,5:PRINT #6;": O O :"
5470 RETURN
5500 REM *** CLEAR MESSAGE WINDOW
5510 FOR Y=2 TO 8 STEP 2
5520 POSITION 10,Y
5530 PRINT #6;"{9 SPACES}"
5540 NEXT Y
5550 RETURN
6000 REM *** FRIEND'S GAMES
6010 GOSUB 2010:REM * FRIEND ASKS CHI
     LD: PLAY A GAME?
6020 ANSWER=1:GOSUB 3210:REM * GET CH
     ILD'S ANSWER
6030 IF M$(1,1)="N" THEN 6080
6040 IF M$(1,1)<>"Y" THEN N=N-1:GOTO
     6010
6050 GOSUB 6110:GOTO 6090:REM * SELEC
     T GAME
6080 N=4:DATNUM=13000:REM * GOOD-BYE
6090 GOSUB 2010:REM * FRIEND SAYS GOO
     D-BYE!
6095 RETURN
6100 REM *** SELECT GAME
6110 DATNUM=12000:N1=N:N=1:REM * RESE
     T DATA POINTERS
6120 GOSUB 2010:REM * GENIE BEGINS GA
     ME-SELECTION QUESTION
6130 READ GAMENUM
6140 N=N+1
6150 FOR Z=1 TO GAMENUM
6160 GOSUB 2010:REM * DISPLAY GAME NA
     ME
6170 GOSUB 3210:REM * GET CHILD'S ANS
     WER
6180 IF M$(1,1)="Y" THEN GOSUB 6310:G
     OSUB 6410:GOTO 6220
6190 IF M$(1,1)<>"N" THEN N=N-1:GOTO
     6160
6200 NEXT Z
6210 DATNUM=12000:N=53:RETURN :REM *
     NO GAMES SELECTED/GOOD-BYE!
6220 DATNUM=13000:N=4:REM * GOOD-BYE!
     :RETURN
6300 REM *** PREPARE FRIEND'S MEMORY
     FOR EXIT FROM FRIEND PROGRAM
6301 REM *** STORE CHILD'S NAME
6302 REM *** IN LOCATIONS
6303 REM *** 1781-1789
6304 REM *** (LENGTH OF NAME IN 1790
6305 REM *** AND SET LOCATION 1791
6306 REM *** AS FLAG THAT
6307 REM *** FRIEND HAS ALREADY
6308 REM *** BEEN CALLED SINCE
6309 REM *** TURNING ON COMPUTER
6310 REM
6315 FOR I=1 TO LEN(NAME$)
6320 POKE 1780+I,ASC(NAME$(I,I))
6330 NEXT I
6335 POKE 1790,LEN(NAME$)
6340 POKE 1791,1
6350 RETURN
6400 REM *** TAPE VERSION OF FRIEND
6410 DATNUM=13000:N=1
6420 GOSUB 2010:REM * ASK IF CHILD NE
     EDS HELP
6430 ANSWER=1:GOSUB 3210:REM * SET CH
     ILD'S ANSWER
6440 IF M$(1,1)="N" THEN 6500
6450 IF M$(1,1)<>"Y" THEN 6410
6460 GOSUB 2010:REM * TELL CHILD HOW
     TO LOAD TAPE/REPEAT STEPS?
6470 N=2:GOTO 6430
6500 RETURN
7000 REM *** FRIEND CALLED ON BEFORE
7010 FOR I=1 TO PEEK(1790)
7020 NAME$(LEN(NAME$)+1)=CHR$(PEEK(17
     80+I))
7030 NEXT I
7040 GOSUB 5010:GOSUB 5210:GOSUB 5460
     :REM * DRAW FRIEND
7050 DATNUM=11000:GOSUB 2010:REM * NE
     W FRIEND MESSAGES
7060 DATNUM=10000:N=3
7070 RETURN
10000 REM *** WAKE-UP FRIEND
10005 REM *** MESSAGES
10010 DATA 3
10011 DATA HI, I'M, GED,-1
10012 DATA YOU,TURNED,ME,ON,-1
10013 DATA WHO'S,OUT,THERE?,-1
10020 DATA 2
10021 DATA I'M,SO,HAPPY,-1
10022 DATA TO,SEE,YOU,*,-1
10030 DATA 2
10031 DATA DO,YOU,WANT,-1
10032 DATA TO,PLAY,A,GAME?,-1
11000 REM *** FRIEND ALREADY AWAKE ME
      SSAGES
11010 DATA 5
11011 DATA HI,*,-1
11012 DATA I,HOPE,YOU,-1
11013 DATA HAD,FUN!!,-1
11014 DATA I,WONDER,WHAT,-1
11015 DATA WE,SHOULD,DO,NOW.,-1
12000 REM *** GAMES
12001 REM
12002 REM *** LIST GAMES ON
12003 REM *** EVERY 10TH LINE--
12004 REM *** LINES 12030-12520
12005 REM *** FOR A MAXIMUM OF
12006 REM *** 50 GAMES.
12007 REM
12010 DATA 2
12011 DATA DO,YOU,WANT,-1
12012 DATA TO,PLAY,-1
12020 DATA 1
```

```
12030 DATA 1
12031 DATA THE,STORY,GAME?,-1
12530 DATA 6
12531 DATA *,I,AM,SORRY,-1
12532 DATA NONE,OF,THE,GAMES,-1
12533 DATA LOOKED,FUN.,-1
12534 DATA MAYBE,WE,CAN,-1
12535 DATA PLAY,LATER.,-1
12536 DATA BYE!,BYE!,BYE!,-1
13000 REM *** MESSAGE TO HELP
13005 REM *** CHILD LOAD GAME
13006 REM *** TAPE
13010 DATA 3
13011 DATA ALL,MY,GAMES,-1
13012 DATA ARE,STORED,ON,TAPE.,-1
13013 DATA DO,YOU,NEED,HELP?,-1
13020 DATA 12
13021 DATA FIRST,PUT,THE,GAME,-1
13022 DATA TAPE,ON,THE,RECORDER.,-1
13023 DATA SECOND,REWIND,THE,TAPE.,-1
13024 DATA THIRD,TYPE,'CLOAD',THEN,-1
13025 DATA PRESS,THE,'RETURN',BUTTON.
     ,-1
13026 DATA FOURTH,PRESS,THE,'PLAY',-1
13027 DATA BUTTON,ON,THE,RECORDER.,-1
13028 DATA LAST,PRESS,'RETURN',AGAIN.
     ,-1
13029 DATA WHEN,THE,PROGRAM,IS,-1
13030 DATA LOADED,TYPE,'RUN'.,-1
13031 DATA WANT,ME,TO,REPEAT,-1
13032 DATA THE,STEPS?,-1
13040 DATA 1
13041 DATA BYE!,BYE!,BYE!,-1
```

©

This series concludes with a comparison of Atari and Apple PILOT instructions, the program variables within the languages, and a table summarizing PILOT commands. The Turtle PILOT language, a version for the VIC and the PET, will be published next month. Last month the Atari version of the language was published and there was a version for the Apple in the September issue.

Turtle PILOT

Alan W. Poole
Loomis, CA

Welcome back for the third and final article about Turtle PILOT. If you haven't seen the last two articles, I would suggest you do that now, unless you are willing to miss out on a powerful new language for your Apple. [*The Atari version of PILOT was published in the October 1982 issue of* **COMPUTE!**, *and the language for the Apple appeared in the September issue.*] In Parts I and II we dealt with all the commands and instructions in Turtle PILOT. Translating an Atari PILOT program to Turtle PILOT will be the main topic of this article, along with a few miscellaneous notes and some documentation on the Editor and Translator programs. At the end of this article you will find a summary of all the commands and instructions in Turtle PILOT.

The program we are going to convert from Atari PILOT to Turtle PILOT first appeared in David Thornburg's "Friends Of The Turtle" column in the April 1982 issue of **COMPUTE!** magazine. Both versions of the program are listed below, along with a list of the changes made to the program to convert it to Turtle PILOT.

1. Line 3 was added to clear the screen and place the cursor in the text window below the high resolution graphics.

2. The symbol placed at the end of a Type instruction to continue the next printed character on the same line is the ampersand instead of the backslash used in Atari PILOT.

3. Turtle commands are preceded by a G: instruction in Turtle PILOT instead of the GR: used in Atari Pilot.

4. Numeric variable names are not preceded by a pound sign (#) in Turtle PILOT, except in a Type instruction, where they are preceded and followed by a pound sign.

5. White was the only color used in the Turtle PILOT version because colors are plotted only at every other coordinate on the Apple.

	Turtle PILOT version	Atari PILOT version
1	*VISITURT	*VISITURT
2	U:*ERASE	U:*ERASE
3	B:HOME:VTAB 21	
4	T:WELCOME TO THE VISIBLE TURTLE	T:WELCOME TO THE VISIBLE TURTLE
5	T:	T:
6	J:*STARTHERE	J:*STARTHERE
7	*MASTERLOOP	*MASTERLOOP
8	T:TURN &	T:TURN \
9	U:*ACCEPT	U:*ACCEPT
10	G:PEN ERASE	GR:PEN ERASE
11	U:*TURTLE	U:*TURTLE
12	G:TURN A	GR:TURN #A
13	*STARTHERE	*STARTHERE
14	G:PEN WHITE	GR: PEN YELLOW
15	U:*TURTLE	U:*TURTLE
16	T:DRAW &	T:DRAW \
17	U:*ACCEPT	U:*ACCEPT
18	G:PEN ERASE	GR:PEN ERASE
19	U:*TURTLE	U:*TURTLE
20	G:PEN WHITE	GR:PEN RED
21	G:DRAW A	GR:DRAW #A
22	G:PEN WHITE	GR:PEN YELLOW
23	U:*TURTLE	U:*TURTLE
24	J:*MASTERLOOP	J:*MASTERLOOP
25	E:	E:
26	*ERASE	*ERASE
27	G:GOTO 0,0; TURNTO 0; CLEAR	GR: GOTO 0,0; TURNTO 0; CLEAR
28	U:*TURTLE	U:*TURTLE
29	E:	E:
30	*ACCEPT	*ACCEPT
31	A:A	A:#A
32	M:E	M:E
33	UY:*ERASE	UY:*ERASE
34	E:	E:
35	*TURTLE	*TURTLE
36	G:GO 4; TURN -90; GO 1; TURN 180	GR: GO 4; TURN -90; GO 1; TURN 180
37	G:30(DRAW 2; TURN 12)	GR:30(DRAW 1; TURN 12)
38	G:GO 1; TURN 180	GR:GO 1; TURN 180
39	G:10(DRAW 2; TURN 36)	GR:10(DRAW 1; TURN 36)
40	G:TURN 90; GO -4	GR:TURN 90; GO -4
41	E:	E:

Miscellaneous Notes On Turtle Pilot

1. It is not necessary to include the asterisk in front

of a label used in the object of a Jump or Use instruction.

2. A Jump instruction without an object will jump back to the last Accept instruction.

3. One dollar sign or pound sign may be used in a Type instruction, but don't use two unless a variable is being Typed.

4. All spaces on the left side of the colon in every instruction will be ignored when the program is translated. Spaces on the right of the colon will be ignored with the G: instruction.

5. When you first run a program, the turtle will be automatically initialized to the middle of the screen, an angle of zero, and the color white.

Program Documentation

So far I have not provided any documentation for the Editor and Translator programs other than the REMarks included in the programs. Lines 50000 to 55060 do not contain any REMarks. These lines are never executed while the Translator is RUNning. Some PILOT instructions require a subroutine to perform the same task in BASIC. This is the purpose of the lines greater than 50000. These lines are included in every translated program. Below is a list of the functions of these subroutines and lists of the variables used in the Editor and Translator programs.

Lines 50000-50050: Initialization. Lines 50020-50025 set the pitch values for notes. Lines 50030-50040 POKE a machine language sound routine into memory.

Lines 51000-51130: Type instruction. QT$ is string to be Typed.

Lines 52000-52220: Accept instruction. Uses the GET command to allow any character to be typed without an error occurring. QI$ is string typed.

Lines 53000-53030: Match instruction. Q$(25) is list of items to be Matched.

Lines 54000-54020: TURN command. QT is number of degrees to turn.

Lines 55000-55060: DRAW command. QL is length of line.

Editor Variables

BELL$: CTRL-G
C: Editor command number.
C$(9): List of Editor commands.
D$: CTRL-D.
F: Temporary flag.
FL: First line number.
I$: General input.
IN$(11): List of PILOT instructions.
K$: Key pressed.
K: ASCII code of K$.
L, L1, L2: Temporary loop counters.
LL: Last line number.
LN: Number of last line in program.
LT: Number of line being typed.
N$: Name of program to be LOADed.
P: Parentheses counter.
P$(2500): Program lines.
RE$: "LINE NO. OUT OF RANGE"
SE$: "SYNTAX ERROR"
T, T$: Temporary variables.

Translator Variables

C: Conditioner, 0 = none, 1 = Y, 2 = N.
D$: CTRL-D.
EX$: Expression.
F: Flag to indicate if turtle commands are in a loop.
GC: Number of turtle graphics command.
G$(6): List of turtle commands in one line of PILOT program.
I: Instruction number.
I$: Line of PILOT program input from disk.
I$(12): List of PILOT instructions.
K$: Character read from PILOT program on disk.
L$: Part of instruction left of colon.
LN: Line number being translated.
LN$: Translated line.
M$(25): Items in object of Match instruction.
N$: Name of program being translated.
NL: Number of lines in PILOT program.
P$(2500): PILOT program lines.
R$: Part of instruction right of colon.
T1$: T$ with spaces removed.
L, T, T$, 11: Temporary variables.

Now that you have learned all the commands and features of Turtle PILOT, you can start writing programs and experimenting with the language. I'm sure you will find that the power and simplicity of Turtle PILOT far outweigh the inconvenience of having to wait a couple of minutes while your program is translated to Applesoft.

Summary Of Turtle PILOT

Editor Commands

ADD – Start or continue program.
LIST – List program.
EDIT – Change line(s) of program.
INSERT – Add a line.
DEL – Delete line(s) of program.
NEW – Erase program in memory.
LOAD – Load program from disk.
SAVE – Save program on disk.
MEM – Display free bytes.
CAT – Catalog disk.
PR# – Change output slot.
ESC – Key to exit Editor.

PILOT Instuctions

- **T:** Type
- **A:** Accept
- **M:** Match
- **J:** Jump
- **U:** Use
- **E:** End
- **C:** Compute
- **R:** Remark
- **S:** Sound
- **G:** Turtle graphics
- **B:** BASIC commands
- * – Used in front of label.
- **Y** – Yes conditioner.
- **N** – No conditioner.

Turtle Commands

- **CLEAR** – Turn on and erase hi-res graphics.
- **TURN** – Add to turtle's angle.
- **TURNTO** – Reset turtle's angle.
- **DRAW** – Draw line at angle turtle is headed.
- **GO** – Move turtle without drawing.
- **PEN** – Change color.
- **SCREEN** – Clear screen to a color.
- **GOTO** – Change coordinates of turtle.
- **FULL** – Full screen graphics.
- **MIX** – Mixed graphics and text.
- **QUIT** – Return to text mode.

©

A Monthly Column

The Beginner's Page

Arrays

Richard Mansfield
Senior Editor

One way to make your computer useful around the house is to have it take over some chores. It could easily do the work of your alarm clock, appointment book, address book, or yearly calendar. What's more, it could combine two jobs into one. To see how one program could do two things at once, let's construct a program which combines the functions of a calendar with an address book. We can also look at how *arrays* work. They can be a most valuable programming tool when you are working with lists. Because Atari BASIC follows nonstandard rules in the construction of arrays, it will not be covered here. The discussion below applies to Microsoft BASIC, which includes Apple, VIC, PET, and many other computers.

When two lists are related to each other, you can put them into a multi-dimensional array and they can work together to provide information. Arrays have *dimensions,* usually one or two. A one-dimensional array is just another word for a common list. In our example program, there is a list of first names (Mary, Bob, Joe, Alice, Mindy) which could be a one-dimensional array. Arrays give the same variable name to a group or list of items. You might think of an array as a massive variable.

The DIM statement creates an array. Line 100 shows how you can DIM an array called A$ to contain "N" by eight items. This is a two-dimensional array. Since we want to allow for expanding the list of people (and their addresses and birthdays), we use the variable N at the start of the program which can easily be changed to show the total.

So, we presently have five people, each listed in a separate DATA statement. The way we are setting up this array, we allow eight pieces of information about each person. These categories are listed in the REM statement in line 5. Imagine a honeycomb of little boxes like those on walls in the post office. Think of each row being used by just a single person. In our DIM, the computer is instructed to set aside enough memory to build five rows. And each row has to be wide enough to hold eight boxes. This is an N X 8 array.

1st Name	Last Name	Street	City	State	Zip	Month	Day
Mary	Jones	15 AL					
Bob	Riley	37 RE					
Joe	Cargile						
Alice	Somme						
Mindy	Clorox						

If we didn't have arrays to work with, we would have to give a different variable name to each item of data. That would mean 40 variable names in this small example program. If you put your own address book into the DATA statements here (and change N in line 5 to equal the total number of people), you might end up with 400 or more pieces of information. Clearly, it is more practical to call Bob Riley's last name A$(2,2) than to give it a unique variable name.

What's more, we can now easily use this array, this expanded A$, as part of a loop. (Keep in mind that A$( ) is not the same as A$. You can use both of them in a program and they won't interact; they are two entirely separate variables.) Notice how lines 100-120 quickly and easily fill up the honeycomb of the array, putting each item into its slot. Then, depending on what kind of information we are requesting – addresses or birthdays – X will point to the second or seventh column in the array. X will let us search through the last names or through the months to find the information desired.

Arrays can make it very easy to solve certain kinds of programming problems. How fast could we change our program to tell us all the people who lived in a particular state? Just change line 300 to:

```
300 ?" WHAT STATE": X=5
```

and the array will be searched for matches in the "states boxes," column five. How would you get information on matching zip codes? You can quickly change the entire function of this program by

AARDVARK – THE ADVENTURE PLACE

ADVENTURES FOR OSI, TRS-80, TRS-80 COLOR, SINCLAIR, PET, VIC-20

ADVENTURES – Adventures are a unique form of computer game. They let you spend 30 to 70 hours exploring and conquering a world you have never seen before. There is little or no luck in Adventuring. The rewards are for creative thinking, courage, and wise gambling – not fast reflexes.

In Adventuring, the computer speaks and listens to plain English. No prior knowledge of computers, special controls, or games is required so everyone enjoys them – even people who do not like computers.

Except for Quest, itself unique among Adventure games, Adventures are non-graphic. Adventures are more like a novel than a comic book or arcade game. It is like reading a particular exciting book where you are the main character.

All of the Adventures in this ad are in Basic. They are full featured, fully plotted adventures that will take a minimum of thirty hours (in several sittings) to play.

Adventuring requires 16k on Sinclair, TRS-80, and TRS-80 Color. They require 8k on OSI and 13k on VIC-20. Sinclair requires extended BASIC.

TREK ADVENTURE by Bob Retelle – This one takes place aboard a familiar starship and is a must for trekkies. The problem is a familiar one – The ship is in a "decaying orbit" (the Captain never could learn to park!) and the engines are out (You would think that in all those years, they would have learned to build some that didn't die once a week). Your options are to start the engine, save the ship, get off the ship, or die. Good Luck.

Authors note to players – I wrote this one with a concordance in hand. It is very accurate – and a lot of fun. It was nice to wander around the ship instead of watching it on T.V.

CIRCLE WORLD by Bob Anderson – The Alien culture has built a huge world in the shape of a ring circling their sun. They left behind some strange creatures and a lot of advanced technology. Unfortunately, the world is headed for destruction and it is your job to save it before it plunges into the sun!

Editors note to players – In keeping with the large scale of Circle World, the author wrote a very large adventure. It has a lot of rooms and a lot of objects in them. It is a very convoluted, very complex adventure. One of our largest. Not available on OSI.

HAUNTED HOUSE by Bob Anderson – This one is for the kids. The house has ghosts, goblins, vampires and treasures – and problems designed for the 8 to 13 year old. This is a real adventure and does require some thinking and problem solving – but only for kids.

Authors note to players – This one was fun to write. The vocabulary and characters were designed for younger players and lots of things happen when they give the computer commands. This one teaches logical thought, mapping skills, and creativity while keeping their interest.

DERELICT by Rodger Olsen and Bob Anderson – For Wealth and Glory, you have to ransack a thousand year old space ship. You'll have to learn to speak their language and operate the machinery they left behind. The hardest problem of all is to live through it.

Authors note to players – This adventure is the new winner in the "Toughest Adventure at Aardvark Sweepstakes". Our most difficult problem in writing the adventure was to keep it logical and realistic. There are no irrational traps and sudden senseless deaths in Derelict. This ship was designed to be perfectly safe for its' builders. It just happens to be deadly to alien invaders like you.

NUCLEAR SUB by Bob Retelle – You start at the bottom of the ocean in a wrecked Nuclear Sub. There is literally no way to go but up. Save the ship, raise her, or get out of her before she blows or start WWIII.

Editors note to players – This was actually plotted by Rodger Olsen, Bob Retelle, and someone you don't know – Three of the nastiest minds in adventure writing. It is devious, wicked, and kills you often. The TRS-80 Color version has nice sound and special effects.

EARTHQUAKE by Bob Anderson and Rodger Olsen – A second kids adventure. You are trapped in a shopping center during an earthquake. There is a way out, but you need help. To save yourself, you have to be a hero and save others first.

Authors note to players – This one feels good. Not only is it designed for the younger set (see note on Haunted House), but it also plays nicely. Instead of killing, you have to save lives to win this one. The player must help others first if he/she is to survive – I like that.

PYRAMID by Rodger Olsen – This is one of our toughest Adventures. Average time through the Pyramid is 50 to 70 hours. The old boys who built this Pyramid did not mean for it to be ransacked by people like you.

Authors note to players – This is a very entertaining and very tough adventure. I left clues everywhere but came up with some ingenous problems. This one has captivated people so much that I get calls daily from as far away as New Zealand and France from bleary eyed people who are stuck in the Pyramid and desperate for more clues.

QUEST by Bob Retelle and Rodger Olsen – THIS IS DIFFERENT FROM ALL THE OTHER GAMES OF ADVENTURE!!!! It is played on a computer generated map of Alesia. You lead a small band of adventurers on a mission to conquer the Citadel of Moorlock. You have to build an army and then arm and feed them by combat, bargaining, exploration of ruins and temples, and outright banditry. The game takes 2 to 5 hours to play and is different each time. The TRS-80 Color version has nice visual effects and sound. Not available on OSI. This is the most popular game we have ever published.

MARS by Rodger Olsen – Your ship crashed on the Red Planet and you have to get home. You will have to explore a Martian city, repair your ship and deal with possibly hostile aliens to get home again.

Authors note to players – This is highly recommended as a first adventure. It is in no way simple – playing time normally runs from 30 to 50 hours – but it is constructed in a more "open" manner to let you try out adventuring and get used to the game before you hit the really tough problems.

ADVENTURE WRITING/DEATHSHIP by Rodger Olsen – This is a data sheet showing how we do it. It is about 14 pages of detailed instructions how to write your own adventures. It contains the entire text of Deathship. Data sheet - $3.95. NOTE: Owners of OSI, TRS-80, TRS-80 Color, and Vic 20 computers can also get Deathship on tape for an additional $5.00.

PRICE AND AVAILABILITY:

All adventures are $14.95 on tape except Earthquake and Haunted House which are $9.95. Disk versions are available on OSI and TRS-80 Color for $2.00 additional.

Please specify system on all orders

ALSO FROM AARDVARK – This is only a partial list of what we carry. We have a lot of other games (particularly for the TRS-80 Color and OSI), business programs, blank tapes and disks and hardware. Send $1.00 for our complete catalog.

AARDVARK - 80
2352 S. Commerce, Walled Lake, MI 48088
(313) 669-3110

Phone Orders Accepted 8:00 a.m. to 4:00 p.m. EST. Mon.-Fri.

TRS-80 COLOR SINCLAIR OSI VIC-20

changing the value of X in line 300. To make all this even more useful, program it to let the *user* decide which column should be searched.

Have your computer put on screen a list of the categories available (change line 5 to PRINT instead of REM) and then INPUT the user's selection. Then use the number that's INPUT as your X. This would be a short step away from a complete, cross-referenced catalog of your library, stamp collection, or whatever. The programming is easy with arrays. The harder part is the time it would take to type in the name and six items of descriptive information about each record in your collection.

```
2 N = 5 : REM THE TOTAL NUMBER OF PEOPLE
5 REM 1ST NAME--LAST--STREET--CITY--STATE--Z
   IP--MONTH--DAY
10 DATA MARY,JONES,15 ALHAMBRA CT.,SLOMO,ARIZ
   ONA,95221,OCT,10
20 DATA BOB,RILEY,37 REVELO DR.,BIXBY,CA,8100
   0,DEC,23
30 DATA JOE,CARGILE,188 S. TATE ST.,NEW YORK,
   NY,10022,APR,11
40 DATA ALICE,SOMMERVILLE,222 DEVLIN,MAXAPAXA
   ,KY,78215,JUNE,15
50 DATA MINDY,CLOROXEUSE,84 MARKMELLO AVE.,SA
   NDY RIDGE,PA,16864,DEC,1
100 DIMA$(N,8):FORI=1TON:FORJ=1TO8
110 READA$(I,J)
120 NEXTJ:NEXTI
130 PRINT" WOULD YOU LIKE TO SEE:  1. ADDRESSE
    S  OR 2. BIRTHDAYS"
140 INPUTK$
150 K=VAL(K$)
160 ONKGOTO200,300
200 PRINT"WHAT IS THE PERSON'S LAST NAME":X=2
210 GOTO400
300 PRINT"WHAT MONTH":X=7
400 INPUTK$
410 FORI=1TON
420 IFA$(I,X)=K$THENQ=1:GOSUB500
430 NEXTI
450 IFQ<>1THENPRINTK$" NOT FOUND. PLEASE CHECK
     SPELLING"
460 Q=0:GOTO130
500 Q=1:FORJ=1TO8:PRINTA$(I,J):NEXT:PRINT:RETU
    RN
```

©

Why use other computer media when you could be using

MEMOREX

high quality error free media?

Free Memorex Mini-Disc Offer – Get free discs!

You'll save money when you buy Memorex, because every carton of 10 Memorex 5¼ inch mini-discs sold by Communications Electronics has a coupon good for a free Memorex mini-disc. For every case of 100 Memorex mini-discs you buy from CE, you'll get 10 free Memorex mini-discs, directly from Memorex. The more you order, the more you save. Offer expires December 31, 1982. All Memorex flexible discs sold by CE are of the highest quality, certified 100% error free and backed by a full one year factory warranty.

Flexible Disc Quantity Discounts Available

Memorex Flexible Discs are packed 10 discs to a carton and 10 cartons to a case. Please order only in increments of 100 units for quantity 100 pricing. We are also willing to accommodate your smaller orders. Quantities less than 100 units are available in increments of 10 units at a 10% surcharge. **Quantity discounts** are also available. Order 500 or more discs at the same time and deduct 1%; 1,000 or more saves you 2%; 2,000 or more saves you 3%; 5,000 or more saves you 4%; 10,000 or more saves you 5%; 25,000 or more saves you 6%; 50,000 or more saves you 7% and 100,000 or more discs earns you an 8% discount off our super low quantity 100 price. Almost all Memorex Flexible Discs are immediately available from CE. Our warehouse facilities are equipped to help us get you the quality product you need, when you need it. If you need further assistance to find the flexible disc that's right for you, call the Memorex flexible disc compatibility hotline. Dial toll-free 800-538-8080 and ask for the *flexible disc hotline* extension 0997. In California dial 800-672-3525 extension 0997. Outside the U.S.A. dial 408-987-0997 between 9 AM to 4 PM Pacific Time.

SAVE ON MEMOREX FLEXIBLE DISCS Product Description	Part #	CE quant. 100 price per disc ($)
8" SSSD IBM Compatible (128 B/S, 26 Sectors)	**3062**	2.09
8" SSSD Shugart Compatible, 32 Hard Sector	**3015**	2.09
8" SSSD CPT 8000 Compatible, Soft Sector	**3045**	2.99
8" SSDD IBM Compatible (128 B/S, 26 Sectors)	**3090**	2.74
8" DSDD Soft Sector (Unformatted)	**3102**	3.34
8" DSDD Soft Sector (128 B/S, 26 Sectors)	**3115**	3.34
8" DSDD Soft Sector (256 B/S, 26 Sectors)	**3103**	3.34
8" DSDD Soft Sector (512 B/S, 15 Sectors)	**3114**	3.34
8" DSDD Soft Sector (1024 B/S, 8 Sectors)	**3104**	3.34
5¼" SSDD Soft Sector w/Hub Ring	**3481**	2.34
5¼" SSDD 10 Hard Sector w/Hub Ring	**3483**	2.34
5¼" SSDD 16 Hard Sector w/Hub Ring	**3485**	2.34
5¼" DSDD Soft Sector w/Hub Ring	**3491**	3.09
5¼" DSDD 10 Hard Sector w/Hub Ring	**3493**	3.09
5¼" DSDD 16 Hard Sector w/Hub Ring	**3495**	3.09
5¼" SSQD Soft Sector w/Hub Ring (96 TPI)	**3504**	2.99
5¼" DSQD Soft Sector w/Hub Ring (96 TPI)	**3501**	3.99

SSSD = Single Sided Single Density; SSDD = Single Sided Double Density; DSDD = Double Sided Double Density; SSQD = Single Sided Quad Density; DSQD = Double Sided Quad Density; TPI = Tracks per inch.

Special offer on Memorex computer tape.

If you mail your order to us and enclose prepayment, deduct $1.00 per reel from our quantity 100 prices. This means Memorex 25JW can be as low as $12.99 in 100 quantities. Memorex Computer Tapes are packed 10 tapes to a carton. Please order only in increments of 100 units for quantity 100 pricing. Quantities less than 100 units are available in increments of 10 units at a 10% surcharge. **Quantity discounts** are also available. Order 500 or more tapes at the same time and deduct 1%; 1,000 or more saves you 2%; 2,000 or more saves you 3%; 3,000 or more saves you 4%; 4,000 or more saves you 5% and 5,000 or more tapes earns you a 6% discount off our super low quantity 100 price. If you need further assistance or information to find the tape that's right for you, call the Memorex Computer Tape Technical Support Group at (408) 987-2937.

SAVE ON MEMOREX COMPUTER TAPE Product Description	Part #	CE quant. 100 price per reel ($)
Memorex IV 2400 feet Wrightline Seal	**25JW**	13.99
Memorex IV 2400 feet Easy Load II Cartridge	**25JR**	14.99
Memorex IV 1200 feet Wrightline Seal	**25FW**	10.65
Memorex Quantum 2400 feet Wrightline Seal	**27JW**	16.20
Memorex Quantum 2400 feet Easy Load II Cart.	**27JR**	16.99
Memorex Quantum 1200 feet Wrightline Seal	**27FW**	12.50
Memorex Cubic HD 2400 feet Wrightline Seal	**39JW**	18.99
Memorex Cubic HD 2400 feet Easy Load II	**39JR**	19.99
Memorex Cubic HD 1200 feet Wrightline Seal	**39FW**	13.99

New Memorex Lifetime Rigid Disc Pack Product Warranty

All Memorex disc packs sold by CE have a lifetime product warranty. This is your assurance that Memorex disc packs will give you a lifetime of performance and service. Only Memorex can offer you the superior reliability of their exclusive M Formula. In addition, Memorex will assist the original user in isolating and correcting any technical issues that relate to the Memorex product as well as, when appropriate, replace up to one set of read/write heads. If you need further information to find the rigid disc that's right for you, call the Memorex rigid disc compatibility hotline. Dial toll-free 800-538-8080 and ask for the *rigid disc hotline* extension 1642. In California dial 800-672-3525 extension 1642. Outside the U.S.A. dial 408-987-1642.

SAVE ON MEMOREX RIGID DISC PACKS Product Description	Part #	CE quant. one price per pack ($)
Mark III 5 MB. Cartridge Front Load (8 to 32 Sect.)	**95-522XX-03**	65.00
Top Load (1-to 24 sectors)	**94-522XX-03**	70.00
CMD-16 "Phoenix Type" CDC Cartridge	**98-26600-31**	160.00
NCR Cartridge	**98-26600-32**	160.00
Mark VIII 80 MB. Error Free	**72-16600-03**	330.00
Flag Free	**72-26600-03**	320.00
Mark XI 200 MB. Error Free	**03-35041**	720.00
Flag Free	**03-35031-02**	560.00
DEC Flag Free	**03-35031-03**	560.00
Mark XII 200 MB. NCR/CDC Flag Free	**03-39001-01**	515.00
Honeywell Flag Free	**03-39000-01**	515.00
Mark XIII 300 MB. Error Free	**03-47021**	795.00
Flag Free	**03-47009**	670.00
Mark XIV 80 MB. Unformated Error Free	**74-16600-03**	365.00
Flag Free	**74-26600-03**	300.00
Honeywell Format Flag Free	**74-26600-08**	315.00
CDC Format Flag Free	**74-26600-09**	315.00
Mark XV 300 MB. Error Free	**03-49011**	825.00
Flag Free	**03-49001-01**	725.00

Smith-Corona TP-1 Letter Quality Printer Special Offer

Buy any Memorex product on this page, and get a Smith-Corona TP-1 letter quality printer for only $585.00 plus $20.00 shipping. Specify serial or parallel version.

Buy with Confidence

To get the fastest delivery from CE of your Memorex computer products, send or phone your order directly to our Computer Products Division. Be sure to calculate your price using the CE prices in this ad. Michigan residents please add 4% sales tax. Written purchase orders are accepted from approved government agencies and most well rated firms at a 30% surcharge for net 30 billing. All sales are subject to availability, acceptance and verification. All sales are final. Prices, terms and specifications are subject to change without notice. Out of stock items will be placed on backorder automatically unless CE is instructed differently. Minimum *prepaid* order $50.00. Minimum *purchase order* $200.00. International orders are invited with a $20.00 surcharge for special handling in addition to shipping charges. All shipments are F.O.B. Ann Arbor, Michigan. No COD's please. Non-certified and foreign checks require bank clearance.

For **shipping charges** add $8.00 per case or partial-case of 100 8-inch flexible discs or $6.00 per case or partial case of 100 5¼-inch mini-discs. For tape shipping, add $1.00 per reel. For Disc packs add $10.00 per cartridge (Mark III or CMD-16) or $15.00 per disc pack for U.P.S. *ground* shipping and handling in the continental U.S.A.

Mail orders to: Communications Electronics, Box 1002, Ann Arbor, Michigan 48106 U.S.A. If you have a Master Card or Visa card, you may call and place a credit card order. Order toll-free. Dial 800-521-4414. If you are outside the U.S. or in Michigan, dial 313-994-4444. Order your Memorex computer products from CE today.

 Ad #090282

MasterCard VISA MEMBER DIRECT MAIL/MARKETING ASSOCIATION CODE OF ETHICS

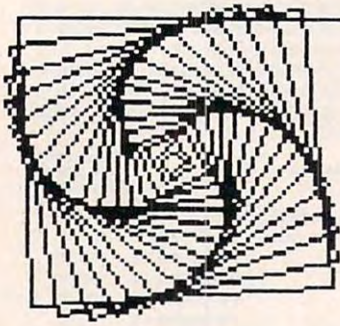
Order Toll-Free!
(800) 521-4414
In Michigan (313) 994-4444

MEMOREX
COMPUTER TAPE
DISC CARTRIDGES · RIGID DISCS
FLEXIBLE DISCS

COMMUNICATIONS ELECTRONICS™

Computer Products Division

854 Phoenix ☐ Box 1002 ☐ Ann Arbor, Michigan 48106 U.S.A.
Call TOLL-FREE (800) 521-4414 or outside U.S.A. (313) 994-4444



A Monthly Column

Friends Of The Turtle

David D. Thornburg
Associate Editor

Recursion – Part I

I saw a comic strip recently that showed a sleeping cat having a dream. The dream was of a sleeping cat having a dream, and so on. The final sleeping cat was dreaming of food. This dream of a dream of a dream is an example of recursion.

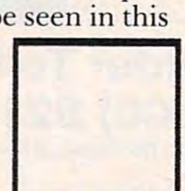
In computer languages, recursion can take several forms. Recursion is probably the most powerful and least understood programming tool in existence. Because LOGO is a marvelous language for exploring this topic, *and* because recursion can let you generate some beautiful pictures with programs only a few lines long, I have decided to devote this and next month's columns to this topic.

The philosophy behind this treatment of recursion is based on my forthcoming book (tentatively titled *Discoveries of Beauty*, Addison-Wesley, 1983) that will be appearing in your neighborhood bookstores very soon.

If you have LOGO on an Apple, TI, or Radio Shack computer, you will be able to experiment with the topics covered in this month's column. If you do not yet have LOGO, this column may help you make a decision to get it.

What is recursion in computer programming? Recursion is the process by which a procedure can use itself repetitively. The simplest type of recursion (supported by every computer language I have ever used) is called *tail-end recursion*. A simple example of tail-end recursion can be seen in this procedure for drawing a square:

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



If you entered this procedure and then started it by typing SQUARE, the turtle would move forward 40 units, turn right by 90 degrees, and then use the SQUARE procedure again. Each time the procedure is used, the turtle adds one more side to the square. After the turtle has drawn four sides, the square is finished, but the turtle will keep retracing its path until you interrupt the program (or until your version of LOGO decides it can't keep track of multiple uses of SQUARE any more).

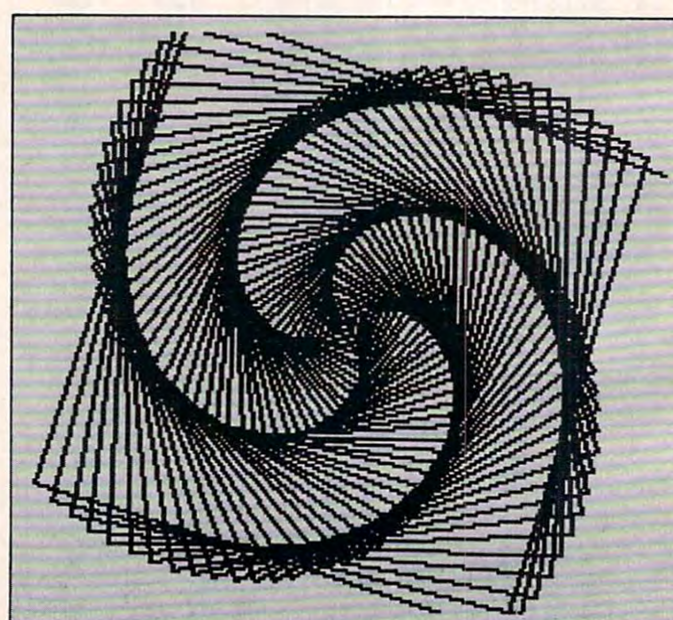
This type of tail-end recursion is available in languages like PILOT through the use of the jump (J:) command, or in BASIC through the GOTO command. The equivalent SQUARE procedure in Atari PILOT looks like this:

```
*SQUARE
GR: DRAW 40
GR: TURN 90
J: *SQUARE
E:
```

Tail-end recursion can also be used to create figures that grow. For example, if we create the LOGO procedure SQUIRAL by entering:

```
TO SQUIRAL :SIZE
FORWARD :SIZE
RIGHT 91
SQUIRAL :SIZE + 1
END
```

then when we enter, for example, SQUIRAL 1, the turtle first moves forward by one unit, turns 91 degrees, and then repeats the procedure with the



new value of :SIZE equal to the old value plus one. The reason that variables can be "passed" to procedures this way is that each time a LOGO procedure is used, the contents of the variables are maintained locally to that use of the procedure. LOGO provides the internal bookkeeping to insure that the value of :SIZE in the second use of SQUIRAL is kept apart from the value of :SIZE we started with. Local variables are a most important feature of languages like LOGO.

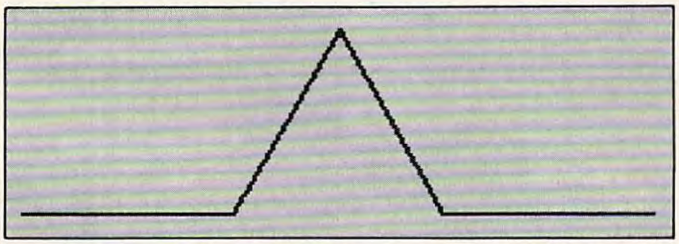
The SQUIRAL procedure also repeats forever, but it does not retrace its own path. This type of tail-end recursion is also possible in languages that have only global (rather than local) variables. In Atari PILOT, for example, this procedure would look like this:

```
*SQUIRAL
GR: DRAW #S
GR: TURN 91
C: #S=#S+1
J: *SQUIRAL
E:
```

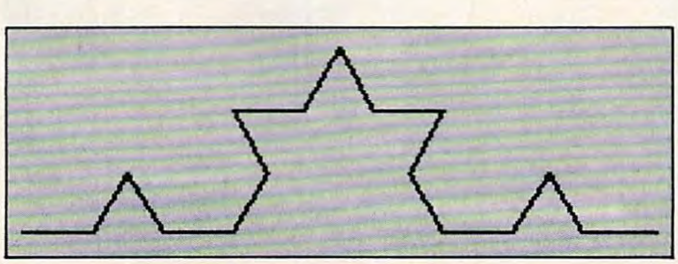
The use of the compute (C:) command allows us to change the value of the numeric variable #S.

As you can see, tail-end recursion is both useful and easy to understand. This form of recursion is just a simple loop from the back of the procedure to the front. Generalized recursion is not so limited.

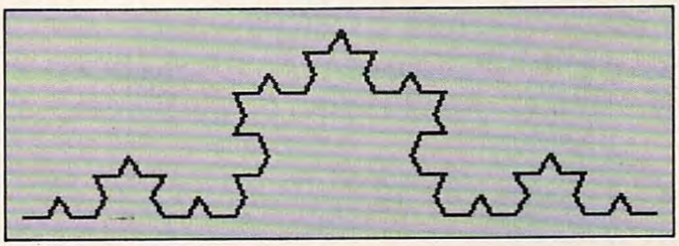
In order to show how general recursion works, we will explore some curves that we described a few columns back – the fractals. Fractal curves are generated by the continued repetition of a simple motif within each portion of an overall curve. For example, suppose we start with the same curve we used in the article on fractals:



By repeating this motif within each straight line segment, we can generate the next pattern in the sequence:



This process can be repeated as many times as we want to get even more complex renditions of the curve:



Explicit Procedures For Drawing Fractals

Before developing a single recursive procedure for drawing this curve, we will explore some explicit methods that will help us understand the recursive form later.

The first procedures we will create are based on the basic triangular bump pattern. To draw this figure, we can use the following two procedures:

```
TO K0 :SIZE
FORWARD :SIZE
END
```

```
TO K1 :SIZE
K0 :SIZE/3
LEFT 60
K0 :SIZE/3
RIGHT 120
K0 :SIZE/3
LEFT 60
K0 :SIZE/3
END
```

(This may appear to be a hard way to draw this figure, but the power of this method will become obvious soon.)

To see the result of this procedure, we should start with the turtle near the left edge of the screen and facing to the right. The following setup procedure should do the job nicely:

```
TO SETUP
PENUP
SETPOS [-120 -60]
RIGHT 90
PENDOWN
END
```

Now enter:

```
CLEARSCREEN
SETUP
K1 243
```

You should see the first level curve on the screen.

We chose 243 for the length of the curve because it fills the screen nicely and because it is a power of three. This last characteristic insures that our more complex renditions of this figure will be drawn with integer line lengths. This is especially valuable for those of you using TI or Radio Shack LOGO.

Suppose we want to draw the next level of this curve. To do this, we need to replace each straight line segment with a replica of the figure generated by K1 with the value of :SIZE reduced by a third. The following procedure does this for us:

```
TO K2 :SIZE
K1 :SIZE/3
LEFT 60
K1 :SIZE/3
RIGHT 120
K1 :SIZE/3
LEFT 60
K1 :SIZE/3
END
```

As you can see, K2 is identical to K1 except that K2 uses the procedure K1, and K1 uses the procedure K0. To see the result of this procedure, enter:

```
CLEARSCREEN
SETUP
K2 243
```

By now it should be pretty clear that we can generate the next level of the Koch curve by creating the procedure:

```
TO K3 :SIZE
K2 :SIZE/3
LEFT 60
K2 :SIZE/3
RIGHT 120
K2 :SIZE/3
LEFT 60
K2 :SIZE/3
END
```

By making a simple modification to K3, we can create the procedure K4 that gives yet another level of detail to our figure, and so on.

How far do we need to continue this process? We could easily create procedures up to K20 or so, but do we really need to? Since our original procedure (K1) drew lines that were 243/3, or 81 units long, then the lines drawn by K2 were 27 units long. K3 used nine unit lines, K4 used three units and, if we were to define it, K5 would use lines one screen unit long. Since the computer display screen can't handle lines less than one unit long, it hardly makes sense to try to create this curve with any more resolution than that.

Because of LOGO's ability to use recursion, we will be able to create a single compact procedure that represents this type of curve to any level of accuracy we wish.

Recursive Procedures For Drawing Fractals

If we look at the procedure K0 through K4, we can see a clue that will show us how to create these fractal curves with only one procedure. The first thing to notice is that K0 is the only procedure that actually draws any lines. The other procedures

Table of Command Sequences for K2

K2	K1	K0
K1 243		
	K0 81	
		FORWARD 81
	LEFT 60	
	K0 81	
		FORWARD 81
	RIGHT 120	
	K0 81	
		FORWARD 81
	LEFT 60	
	K0 81	
		FORWARD 81
LEFT 60		
K1 243		
	K0 81	
		FORWARD 81
	LEFT 60	
	K0 81	
		FORWARD 81
	RIGHT 120	
	K0 81	
		FORWARD 81
	LEFT 60	
	K0 81	
		FORWARD 81
RIGHT 120		
K1 243		
	K0 81	
		FORWARD 81
	LEFT 60	
	K0 81	
		FORWARD 81
	RIGHT 120	
	K0 81	
		FORWARD 81
	LEFT 60	
	K0 81	
		FORWARD 81
LEFT 60		
K1 243		
	K0 81	
		FORWARD 81
	LEFT 60	
	K0 81	
		FORWARD 81
	RIGHT 120	
	K0 81	
		FORWARD 81
	LEFT 60	
	K0 81	
		FORWARD 81
END OF EXECUTION		

only use lower numbered procedures themselves, or turn the turtle. By writing the actual steps as they are executed, we can show how these procedures work. Let us examine K2, for example. If we expand the steps, we can see the sequence of commands as they are carried out. Each column in the table below shows a different procedure. Since K2 uses K1 and K1 uses K0, this table needs only three columns. The arrows show the direction in which control is passed from one procedure to the other.

When we used K2, it used K1 four times, and K1 used K0 16 times to actually draw the lines. A table for K3 would be four times longer than this and would require four columns. If you decide to construct such a table yourself, you will see that by the time K3 has finished, it will have used K2 four times, K1 16 times, and K0 64 times.

Because of the similarities between K1, K2, K3, etc., we should be able to use one procedure to create these curves with any level of complexity we want. We can do this because when LOGO procedures use themselves recursively, LOGO creates as many new copies of the procedure as are needed to keep the levels uniquely identified.

The only procedure we created that is markedly different from the rest is K0, because it only draws lines. The following procedure incorporates all the features of K0, K1, K2, etc., into one compact form that lets us generate any level of this curve we desire.

```
TO TRIAD :SIZE :LIMIT
IF :SIZE < :LIMIT [FORWARD :SIZE STOP]
TRIAD :SIZE/3 :LIMIT
LEFT 60
TRIAD :SIZE/3 :LIMIT
RIGHT 120
TRIAD :SIZE/3 :LIMIT
LEFT 60
TRIAD :SIZE/3 :LIMIT
END
```

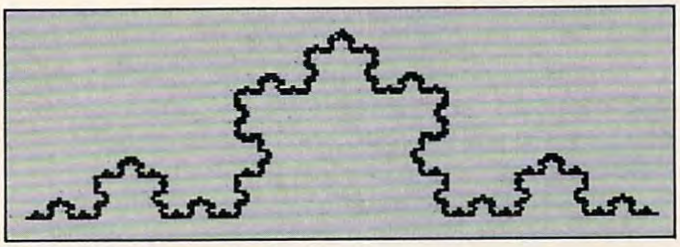
To see how this procedure operates, let's examine it line by line. Suppose you were to give the command TRIAD 243 100, for example. First, the size (243) would be tested to see if it was less than the limit (100). Because it is not, TRIAD would be used again with a size of 243/3, or 81. Since in this new use of TRIAD the size (81) is less than 100, a line will be drawn (just as with the procedure K0). As soon as this happens, the STOP command forces LOGO back to the earlier version of TRIAD to carry out its next command (LEFT 60). This process is continued in just the same way that K1 used K0. The only difference is that we are taking advantage of LOGO's ability to keep track of multiple uses of a procedure we have defined only once. It is as though LOGO makes as many copies of TRIAD as it needs and gives them and their variables special names so that they are used in the right order and without getting the contents of the variables confused.

Experiment with TRIAD (leaving the turtle visible). By watching the figure being drawn, you might gain more insight into the way that recursion is being used to create the figure. To generate the figures we have already drawn, you might use:

```
TRIAD 243 243
TRIAD 243 81
TRIAD 243 27
```

Remember to clear the screen and use SETUP before drawing each curve. To see the most detailed level of this curve that can be shown on the screen, enter

```
CLEARSCREEN
SETUP
TRIAD 243 3
```



Next month we will continue with more examples of fractal curves and explore a few more complex examples of recursion. In the meantime, please feel free to experiment on your own!

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

©

A look at three Apple and Atari programs which assist the computer artist.

A Monthly Column

Learning With Computers:

Computers In The Art Class

Glenn M. Kleiman
Teaching Tools: Microcomputer Services
Palo Alto, CA

An important first lesson in computer literacy is that computers are flexible tools for working with all types of symbols – numbers, words, pictures, and sounds. Many people think of computers as "number-crunching" machines, useful only for business, mathematics, and science. Since computerized word processing has become popular in the last few years, more people have realized that computers can be used for working with words as well as numbers. But few people are aware of the potential of computers for working with pictures and sounds. In this column, I will focus upon *computer graphics* – the use of computers to create pictures.

Computers already influence our visual environment. Movie makers use computers to produce all sorts of special effects. The best example is the movie *TRON*, which contains superb computer-generated animation that appears to be three dimensional. Pictures generated with computers are used in television shows and commercials, magazine advertisements, stadium scoreboards, and, of course, video games. Computer graphics are becoming widely used in business to produce charts, graphs, and other pictorial representations of the results of number crunching. Artists, architects, designers, cartoonists, engineers, and educators are all using computer graphics.

Personal computers capable of high resolution color displays are powerful tools for computer graphics. You will not be able to fully replicate the images of *TRON*, but you can create all sorts of pictures, even three dimensional animations.

You can create pictures on computer screens by writing programs in BASIC, LOGO, or other languages. However, to really explore computer graphics, you will want a program designed to make it easy to create and manipulate pictures – a graphics editor. As word processing programs facilitate working with written text, graphics editors facilitate working with pictures.

You can use graphics editors to create pictures to be combined with computerized lessons, simulations, or games, to provide visual aids for presentations, and for many other functions. Best of all, you can use these programs to explore this exciting new medium for creativity.

Available graphics editors vary in capabilities, ease of use, necessary hardware, and price. Some are combined with special drawing surfaces, so pictures drawn on the surface are transferred to the computer screen. Others use game paddles, joysticks, light pens, or the keyboard. These editors let you draw pictures quickly and easily and may contain other options for colors, textures, changing sizes, combining pictures, and so on. The following descriptions will give you an idea of how these enjoyable tools can encourage you to explore the exciting world of computer graphics.

The Designer's Toolkit

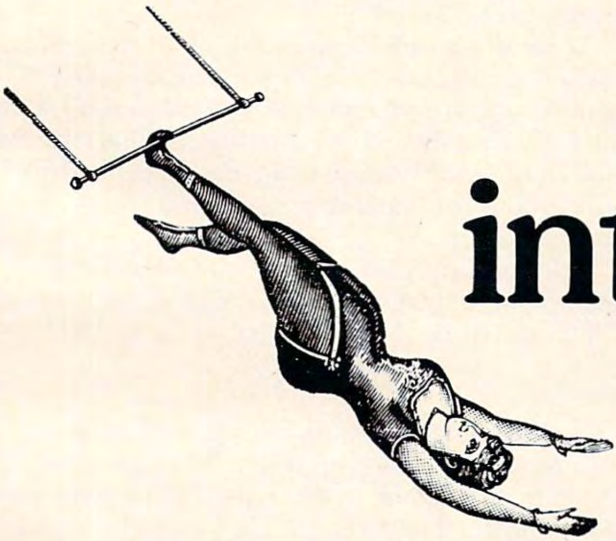
The *Designer's Toolkit* is a top-of-the-line graphics creation program. Although too expensive for most people, it provides a high standard, both in capability and ease of use, against which other programs can be evaluated.

The *Designer's Toolkit* is for Apple II computers equipped with a graphics tablet. A graphics tablet is a thin, flat device, about 16 inches on each side, with a stylus attached by a cable. Through a special interface and software, the computer can decode where on the tablet the stylus is touching and whether or not the tip on the stylus is pressed.

The *Designer's Toolkit* was developed to make all of the graphics capability of the Apple II available and simple to use. The package includes the toolkit disk, a demonstration disk, a 115-page manual with 32 color pictures (all created with the *Designer's Toolkit*), and a plastic overlay to put on the graphics tablet.

Most of the graphics tablet is used as a drawing area, but the top and bottom are used to select options in the program. The overlay contains boxes with colors, shapes, and words. You select each option by touching the stylus to the appropriate box. This lets you use almost all the program's capabilities from the drawing surface, without having to use the keyboard.

The simplest option is drawing. If you hold the stylus near the tablet, a cross mark appears on the screen to show the position that corresponds to the location on the tablet where the stylus is pointed. If you press the stylus down and move it, a line appears. With a little practice, it becomes completely



natural to move the stylus on the tablet while you are watching the screen.

Ten Permanent Brushes

If you were painting, you would not want to be limited to a single thin brush and white paint on a dark background. The *Designer's Toolkit* provides ways to change the "brush," "paint," and background. Ten permanent brushes are built into the program. These vary in thickness and shape. Some are round and others are long and thin, like a calligraphy pen. You can change your brush at any time by simply touching the stylus to the box representing the brush you would like. Similarly, you can select for either the background or paint color any of 16 colors permanently built into the program.

This selection of brushes and colors is very powerful, but it is just beginning. There are also ten user-defined brushes and ten user-defined colors. The package contains special programs for creating your own brush and color sets. You can make a brush of any shape – a person, letter, pattern, or whatever you like. (For those familiar with Apple programing, the set of user-defined brushes is a shape table.) The color definition program lets you select from 160 possible colors, created by combinations of different dot patterns.

If you were drawing on paper, you might use a ruler, protractor, and compass to create straight lines, angles, and circles. There are options for lines, triangles, rectangles, and circles. Each of these options uses a "rubber-band" technique. For example, after you select the rectangle option, you place the stylus where you want one corner of the rectangle. You then slide the stylus to locate the opposite side. As you move the stylus, the rectangle changes size and shape so you can see exactly where it will appear. When you lift the stylus, the rectangle is automatically drawn with the current brush and color.

There are also options to rotate pictures along the horizontal or vertical axes; to fill enclosed areas with any color; to change colors or to remove colors; to relocate pictures on the screen; to pick up a section of the screen and repeat it elsewhere (a "rubber stamp" option); and to define a temporary "window" to restrict changes to one section of the screen (for erasing or coloring one section only).

The Apple II can hold in memory two pages of high resolution graphics at one time, and you can alternate pages displayed on the screen. The *Toolkit* lets you copy pictures from one page to the other and merge pictures. You can make a set of small pictures, save them on a disk, load one picture at a time to one page, slide, invert, rotate, or color it, and then merge it with the other page. This makes it possible to create a complete picture from a set of simple ones. There are three merge options, which let you simply combine pictures (OR merge), combine pictures while erasing any parts that overlap (XOR merge), or create a picture with only the parts of the two pictures that overlap (AND merge).

A Magnify Option

There's more. A fantastic magnify option lets you select a section of a picture and magnify it to be anywhere from two to 64 times as large. The original picture appears on one page, and the magnified image appears on the other. You can then change the magnified image, and the changes automatically appear, in reduced size, on the original! This is ideal for making very detailed drawings and for making careful corrections.

You can also add text, in any of 18 type fonts, anywhere on the screen. There is even a review picture program, which lets you create a slide show of pictures on a disk. The extensive manual explains all the options and contains a great deal of information about the capabilities and limitations of Apple graphics.

In sum, simply amazing. Now the bad news: the graphics tablet costs $800, the *Designer's Toolkit* $225 (both from Apple dealers).

Paint

Paint is a graphics creation program for Atari computers. It requires joysticks; if you are already a *Pac-Man* or *Asteroids* player, you do not need any new hardware. The *Paint* package includes a disk with three programs (Simple Paint, Super Paint, and Art Show) and a 145-page manual.

The Simple Paint program is designed so most three-year-olds could use it successfully. Once the program is started, only the joystick is used – the keyboard is never needed. The bottom of the screen shows four "paint pots," four "brushes," and an "erase" box. The center of the screen shows a marker which can be moved with the joystick. The child can select a paint color by moving the marker to the paint pot and pressing the button on the joystick.

A brush size is selected by moving the marker to one of the brushes and pressing the joystick button. When the marker is moved and the button held down, a line is drawn in the color and brush size chosen. The joystick controls the direction and length of the line. A new color or brush can be chosen at any time. The erase box is for clearing the entire screen. Sections of the screen can be erased by painting over them with the background color.

Simple Paint makes available all the colors the

Atari can display. To change a color, the child moves the marker to a paint pot and presses the button twice. Then, moving the joystick to the left changes the hue; moving it to the right changes the saturation. When the desired color appears, the child presses the button and resumes painting.

Super Paint adds a number of powerful features to those of Simple Paint. Each option can be selected from the keyboard or by using the joystick to display and choose items on menus. There are nine different shapes of brushes, each of which can be in any of nine sizes. There are options for straight lines, circles, and rectangles. To draw a circle, for example, you select the circle option, move the marker to where you want the center, press the button, and then move the marker to anywhere on the circumference of the desired circle. When you press the button again, the computer completes the circle.

The Zoom Option

You can set the speed of the brush movement to draw quickly or slowly. You can fill areas with a color and change one color on the screen to another. You can select paint colors as in Simple Paint, but Super Paint also lets you draw with plaids, stripes, and other patterns.

A "zoom" option magnifies your picture. When the picture is magnified, the screen functions as a window which can be moved to display different sections of the picture. You can draw on the large picture and then shrink it back to its original size. You can save pictures on disks and use the Art Show program to show them as a series of continuous slides.

The main limitations of *Paint* are due to the hardware used. It is more difficult to draw with a joystick than with a stylus on a surface, and the joystick registers only eight different positions, so you can draw only angles in 45 degree increments. Also, in the graphics mode used, the Atari can display only three colors at a time.

Paint is one of the best designed programs I have seen. I have observed children as young as six master most of the options of Super Paint by exploration, with little help from adults. I have also observed a professional watercolor artist who had never before used a computer become fascinated with creating with *Paint.*

The first 45 pages of the 145 page manual describe the programs; the rest is a well-done description of the way computers work, the history of art, the relation of computer art to other art forms and to our culture, the uses of computer graphics, biographies of computer artists, and ideas for using *Paint.* The book is a valuable introduction to computer art even without the program.

The *Paint* package sells for $39.95 (available from Reston Publishing Company, 11480 Sunset Hills Road, Reston, Virginia 22090). Developed at the Capital Children's Museum in Washington, D.C., *Paint* is an outstanding software/book package and an exceptionally good value.

Edu-Paint

Edu-Paint is an inexpensive grahics creation program for the Apple II. It requires game paddles (or a potentiometer-type joystick). You draw with the paddles as if you were using an Etch-A-Sketch. You can choose colors, and there is a "palette" for creating blends and patterns. You can draw lines, circles, and rectangles, fill enclosed areas, and duplicate a section of the screen (like the stamp option in the *Designer's Toolkit*). Each option is chosen from the keyboard. *Edu-Paint* is an easy-to-use graphics creation program. It is available for $10 from Softswap, Microcomputer Center, San Mateo Office of Education, 333 Main Street, Redwood City, CA 94063. For a catalog only and information, send $1 to the same address.

VersaWriter

The *VersaWriter* is a hardware and software package available for Apple II, Atari, and IBM computers. (I have not seen the IBM version.) The hardware is a drawing board with a pointer attached. The computer can decode the position of the pointer on the drawing pad.

The VersaWriter seems designed primarily for transferring pictures from paper to the computer screen. You can place a picture on the drawing board and trace over it with the pointer. The software lets you change the size of the picture as you trace over it. You can draw with several different brushes and with many colors, fill enclosed areas with color, add text to pictures, and select other functions. The software also lets you create shape tables on the Apple II or player/missile shapes on the Atari. Additional "expansion pack" programs are available for the Apple, to magnify or shrink pictures, combine two screens into one picture, and rotate pictures.

The *VersaWriter* is a good tool for creating graphics to incorporate into your own programs. It has the advantage over *Paint* of providing a drawing tablet which allows better control and the advantage over the *Designer's Toolkit* of being less expensive. However, it is not as smooth or quick to use as the *Designer's Toolkit* or *Paint.* You do have to switch between the keyboard and the drawing board for every command, and if you draw quickly the computer doesn't keep up. The *VersaWriter,* therefore, does not encourage creative art work as well as the other packages do. The *VersaWriter* tablet and software package is available for $299.95 from Versa Computing, 3541 Old Conejo Road, Suite 104, Newbury Park, CA 91320.

Versa Computing also markets for Atari computers a less expensive ($39.95) *Graphics Composer* program which uses a joystick instead of the drawing board. Although not as flexible as *Paint* for creating pictures, it contains capabilities (not found in *Paint*) for adding text to pictures and for creating player/missile shapes. Like the *VersaWriter* the *Graphics Composer* seems better designed for creating graphics to incorporate into programs than for exploring computer art.

Why Explore Computer Art?

As Alex Packer, author of the book accompanying the *Paint* program, writes:

> It only seems appropriate that a culture so thoroughly linked to technology and machines should create art with the ultimate machine of our times, the computer. The computer is an artist's tool. Instead of a chisel, a brush, a stick or a trowel, the artist paints with a computer. Instead of oil paints, acrylics, pastels, charcoal or sand, the artist paints with electronics. Instead of canvas, plaster, wood, marble or paper, the artist paints on a cathode ray tube; light is the medium. Throughout history, the breakthroughs of science have been integrated, directly and symbolically, with art forms... Where will it lead? Nobody knows. It will take years to explore the expanded creative flexibility and techniques offered by the computer.

©

This article describes how a communications system can be set up between a PET (Upgrade ROMs) and a Hewlett-Packard HP3000. The problems solved during the creation of this system suggest solutions to other similar peripheral communications tasks.

A Terminal Operating System For PET To HP3000 +

Penny Peterson
Department of Chemistry
Bryn Mawr College
Bryn Mawr, PA

We have developed a Terminal Operating System (TOS) to link a PET (Upgrade ROMs) to a Hewlett-Packard HP3000 Series III computer. The guiding principle was to implement a link that would permit using the PET as a dumb terminal and that would also permit the transfer of programs and data.

In addition to a standard Model 32N PET, our link made use of a SADI from Connecticut Microcomputers Corporation for the needed conversion from PET IEEE-488 Parallel to RS-232-C Serial. A Bell modem and phone lines provided the remote connection to the main computer. A Toolkit ROM supplement provided some useful software features, but led to some modifications.

The TOS described here was part of a larger project to construct a PET-controlled tunable dye laser spectroscopy system with data acquisition and graphics capability. The TOS provided a link to a larger computer for program editing, major file storage, large scale data processing, and routine terminal access. In particular, the TOS has been used to upload programs from the PET to the HP3000 using the Toolkit APPEND function.

General Idea Of TOS

We want the PET to simply transmit data from the keyboard to the HP3000 via the SADI, and display to the screen data received from the HP3000 via the SADI. There are also some compatibility problems between the PET and the HP3000 which the program must compensate for.

The program first opens a file to the SADI using the options: 300 baud, auto suppression of line feeds from the HP3000, reversal of upper/lowercase from the PET, and a lot of nulls to be sent by the SADI to the HP3000 after a Carriage Return. The program also POKEs the appropriate values in memory to select the alternate character set (which includes lowercase) and to enable the cursor (needed for use in the HP3000's line-oriented editor).

The main body of the program consists of two loops – one to get a character from the keyboard and send it to the SADI, and one to get a character from the SADI and send it to the screen. In each loop, any conversions of characters necessary to achieve compatibility are made. (Most of these compatibility problems were not evident until the program was in operation.)

In the loop which gets characters from the keyboard, provisions are made to convert PET keys to control keys for the HP3000. PET DELETE is converted to HP3000 BACKSPACE; PET CURSOR RIGHT is converted to HP3000 CONTROL Y (Software Break for use in the Editor). PET CLEARSCREEN (Shift Home) is used to leave the TOS.

In the loop which gets characters from the HP3000, provisions are made to suppress unwanted characters sent by the HP3000. The HP3000 sends XOFF (Control Q) to indicate the end of a message. This control character is an ASCII 17, which happens to correspond to a PET linefeed. The program does not transmit this character, thus avoiding unwanted extra linefeeds. Another compatibility problem was that the HP3000 sends Carriage Return and Linefeed separately, but the PET automatically does a Linefeed upon receipt of a Carriage Return. This was solved by simply ignoring Linefeeds sent by the HP3000. Finally, the PET's blinking cursor caused problems. If the cursor happened to be blinking "on" just before a Carriage Return, a stray cursor would be left behind while the new cursor moved on to the next line. This was prevented by turning off the cursor before executing a Carriage Return from the HP3000, and turning it back on again afterwards.

Speeding Up The Old TOS

The original version of the TOS worked, albeit with some losses of characters in both directions due to speed problems. The BASIC program in the PET that is the TOS must run very quickly in order to catch all incoming characters from the HP3000. A sure-fire way to make the program run fast enough would be to rewrite it in machine language, but this would require a considerable amount of work.

Therefore, I tried all the tricks at my disposal to make the program run fast enough in BASIC.

This involved removing all REMark statements and putting the loops where speed was crucial at the beginning of the program. These loops made liberal use of GOTOs, and, since the execution of a GOTO requires searching sequentially through the program line numbers until the desired one is found, considerable time can be saved by having line numbers frequently jumped to at the beginning of the program. These modifications speeded up the program noticeably.

At this point I discovered that when the "BASIC Programmer's Toolkit" ROM was invoked (SYS 45056), the revised, speed-conscious TOS was not fast enough. Unfortunately, the only documented way to turn the Toolkit off was to turn off the PET, of course resulting in the loss of any programs in the PET. The Toolkit's APPEND function was needed to append programs to be uploaded onto the end of the TOS. (More about the Uploading capabilities of the TOS later).

So until we figured out a software way to turn the Toolkit off, a complicated and time-consuming series of reads and writes to tape was necessary to upload a program to the HP3000 printing (e.g., Load TOS, Append program to be uploaded, Save the TOS + program combination, Turn off the PET to turn off the Toolkit, Reload the TOS + program combination, Run TOS). Disassembling sections of PET BASIC and the Toolkit ROM uncovered the patch which the Toolkit makes to the BASIC Input/Output routines.

In addition to the normal checks for BASIC keywords, the Toolkit adds checks for the Toolkit keywords. These additional checks also slow down BASIC Input/Output. A machine language routine was written which replaced the Toolkit patch with the original BASIC routine. (See Program 2 machine language routine, written by Gary Kaufman, which turns off the Toolkit.) After this routine was incorporated into the TOS, the PET no longer missed characters coming from the HP3000.

The other half of the speed problem – the HP3000 Editor's loss of characters from the PET – occurred because there is a delay between the time that the Editor accepts a line of input (terminated by a carriage return) and the time it starts accepting the next line. This means that any characters sent from the PET to the HP3000 during the delay will be lost. The use of the SADI's ability to send multiple nulls to the HP3000 after a carriage return is an attempt to send only meaningless information during the Editor delay.

This helps alleviate the problem, but does not totally cure it. This loss of characters by the HP3000 Editor is noticeable only when information is being transmitted very quickly, thus ordinary typing into the Editor is not affected. However, the original Uploading routine, which LISTs information to the HP3000 at about 30 characters per second, was hampered. (See below for instructions for use of original uploading routine.)

Uploading Into The HP3000 Editor

The purpose of the Uploading routine was to transfer a BASIC program from the PET's memory to the HP3000. The original routine works on the premise that the HP3000 Editor neither knows nor cares whether the input it receives from the SADI is being typed in by hand. By using the BASIC commands LIST and CMD, the program to be uploaded can be listed directly to the SADI. First, a file to the SADI must be opened, specifying a non-PETcontroller and conversion of PET graphics characters to printable mnemonics. Then the command CMD is given, which transfers the PET-User screen dialogue to the HP3000. The subsequent LIST command is performed on the new designated device – the HP3000 Editor.

As mentioned before, the accuracy of the uploading is limited by the time lag in the HP3000 Editor that occurs between the receipt of one line of text and the acceptance of the next. This is a major limitation which cannot be overcome. We thought about developing a handshaking protocol in which the HP3000 Editor would signal when it was ready to accept a new line; but this plan was discarded because it would require each line to be listed individually, and the LIST command is not capable of this. (It is essential to use the LIST command because PET BASIC programs are stored in memory in tokenized form, and LIST is one of the few commands which untokenizes.)

Uploading Into A FORTRAN/3000 File

Even if occasional loss of characters for uploading programs could be lived with, it was certainly unacceptable for the uploading of data. Thus, a completely new uploading routine was designed which did not rely on the HP3000 Editor (Program 3). The new routine lists the program to be uploaded directly into a HP3000 data file via a FORTRAN program.

The data file is created using the :BUILD. The file created must be large enough to hold the programs or data to be loaded into it from the PET. A FORTRAN program (INFILE) reads lines in from the keyboard. Again, lines LISTed from the PET are indistinguishable from lines typed in at the keyboard. These lines are stored into the data file (UPLOAD). A separate FORTRAN program (OUTFILE) allows reading of the data file. Since a FORTRAN program will wait for input as long as necessary, there is no problem with lost characters. This Uploading routine even runs

THE BUSINESS MANAGER...

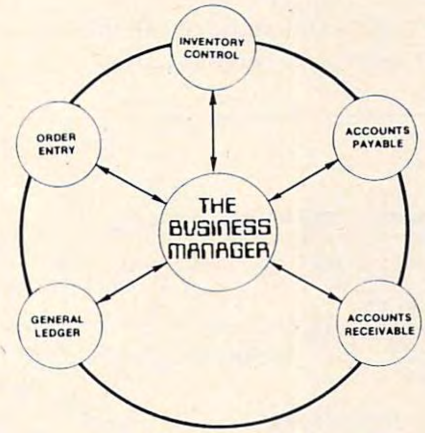
FULLY INTERACTIVE ACCOUNTING SYSTEM. NO DISKETTE CHANGING! FINALLY ALL THE FEATURES YOU'VE BEEN WAITING FOR – IN ONE COMPLETE INTERACTIVE SYSTEM.

FULL SOFTWARE SYSTEM

ONLY $995.00

QUESTAR INTERNATIONAL INC

MANUAL ONLY – $40.00 REFUNDABLE UPON SYSTEM PURCHASE.



HARDWARE REQUIREMENTS:

- Commodore Model 8032 or SuperPET or 8096 Micro-computer
- Commodore Model 8050 One Megabyte Dual Disk Drive
- Commodore Model 4022 (or Larger) Printer
- Commodore PET — IEEE cable
- Commodore IEEE — IEEE cable

DISKSHARE 9000

WORKS WITH ALL COMBINATIONS PET/CBM 100% HARDWARE. NO SOFTWARE REQUIRED! UP TO 8 USERS SHARE DISK AND/OR PRINTER. CONNECTS IN MINUTES – NO TOOLS REQUIRED. INCREASES PRODUCTIVITY – REDUCES COSTS. IDEAL FOR EDUCATION AND BUSINESS. CABLE LENGTHS UP TO 50 FEET.

ONLY $749.00 (U.S.)

8 USER SYSTEM ONLY $1149

Multi User System

DEALER INQUIRIES WELCOME.

QUESTAR INTERNATIONAL INC.
Suite 102, 7270 Woodbine Avenue
Markham, Ontario Canada L3R 1A4
Tel. (416) 475-8044

faster because it does not require the multitude of nulls to be sent after a carriage return.

Program 1.

```
50000 REM TERMINAL OPERATING SYSTEM
50010 REM PENNY PETERSON
50020 REM PET TO HP3000 VIA SADI
50030 REM MENU
50040 POKE59468,12                                    ~
      :REM NORMAL CHARSET
50050 PRINT"(H(C> TOS: T)ERMINAL U)PLOAD Q)UIT":
    PRINT
50060 GETA$
50070 IF(A$="T")GOTO50120
50080 IF(A$="U")GOTO50310
50090 IF(A$="Q")GOTO50380
50100 GOTO50060
50110 REM ACT AS A DUMB TERMINAL
50120 CLOSE 5
50130 PRINT"(H(CPET TO HP3000 TERMINAL OPERATING
     SYSTEM"
50140 POKE 59468,14                          :REM AL
    TERNATE CHARSET
50150 POKE167,0                                       ~
    REM CURSOR ENABLE
50160 D$="EAR"      :REM SADI DESCRIPTORS E-300 B
    AUD; A-AUTO SUPPRESS
50161 :REM  R - REVERSES UPPER/LOWER CASE FROM P
    ET
50170 OPEN 5,4,15,D$       :REM SADI DEVICE #4; S
    ECONDARY ADDRESS = 15
50179 REM
50180 GET A$:IF A$="" THEN 50240                  :
    REM GET CHAR FROM PET
50181 :REM IF NOT FOUND, GO CHECK FOR CHAR FROM ~
    HP
50190 IF ASC(A$)=13 THEN PRINT#5:GOTO50180 :REM ~
    IF <CR> SEND IT TO HP
50200 IF ASC(A$)=20THENA$=CHR$(8)     :REM PET DE
    LETE --> HP BACKSPACE
50210 IFASC(A$)=29THENA$=CHR$(25)           :REM PE
    T CURSOR RT -->CNTL Y
50220 IFASC(A$)=147THEN50030      :REM IF PET CLR
    SCREEN, JUMP TO MENU
50230 PRINT#5,A$;          :REM SEND CHAR FROM PE
    T KEYBOARD TO HP3000
50239 REM
50240 GET#5,A$:IF A$="" THEN 50180                ~
    :REM GET CHAR FROM HP
50241 :REM IF NOT FOUND, GO CHECK FOR CHAR FROM ~
    PET
50250 IFA$=CHR$(17)THEN50180            :REM SUPPR
    ESS UNWANTED LINEFEED
50260 IFA$=CHR$(13)THENPOKE167,1:PRINT" ":A$="":
    POKE167,0:GOTO50180
50261 :REM WIPE CURSORS FROM END OF LINES SO DON
    'T LITTER SCREEN
50270 IFA$=CHR$(10)THENA$="":REM SUPPRESS LINEFE
    ED FROM HP
50280 PRINT A$;                         :REM SEND ~
    HP CHAR TO PET SCREEN
50290 GOTO 50180
50300 REM UPLOAD
50310 CLOSE 5
50320 PRINT"(H(CUPLOADING..."
50330 POKE 167,1                                  :
    REM DISENABLE CURSOR
50340 OPEN 5,4,15,"EPC997"                        :
    REM SADI DESCRIPTORS
50341 :REM 300 BAUD/NONPET CONTROLLER/PRINT CONT
    ROL CHARS/
50342 :REM  9+9+7= 25 EXTRA NULLS SENT AFTER <CR
    >
50350 CND 5                   :REM TRANSFER SCREEN DI
    ALOGUE FROM PET TO HP
50360 LIST              :REM SENDS PROGRAM TO BE UP
    LOADED FROM PET TO HP
50370 REM LIST KICKS US BACK TO BASIC
50380 CLOSE 5
50390 END
```

Program 2.

```
140 PRINT"(H(C"
150 PRINT"TOOLKIT DISCONNECT ROUTINE"
160 N=832:FORI=57647TO57656
170 X=PEEK(I):POKEN,X:N=N+1:NEXTI
180 FORI=826TO831:READX:POKEI,X:NEXTI
190 FORI=842TO848:READX:POKEI,X:NEXTI
200 DATA165,119,72,165,120,72
210 DATA104,133,120,104,133,119,96
220 POKE833,24:SYS826
230 PRINT"TOOLKIT DISCONNECTED"
240 END
```

Program 3.

```
0 GOTO100
1 GOSUB 29
2 GOTO15
3 GETA$:IFA$=""THEN9
4 IFASC(A$)=13THENPRINT#5:GOTO3
5 IFASC(A$)=20THENA$=CHR$(8)
6 IFASC(A$)=29THENA$=CHR$(25)
7 IFASC(A$)=147THEN15
8 PRINT#5,A$;
9 GET#5,A$:IFA$=""THEN 3
10 IFA$=CHR$(17)THEN3
11 IFA$=CHR$(13)THENPOKE167,1:PRINT" ":A$="":
   POKE167,0:GOTO3
12 IFA$=CHR$(8)THENA$=CHR$(20)
13 PRINTA$;:GOTO3
14 RETURN
15 POKE59468,12:PRINT"(H(C> TOS: T)ERMINAL U)
   PLOAD Q)UIT"
16 GETA$:IF(A$="T")GOTO21
17 IF(A$="T")GOTO21
18 IF(A$="U")GOTO24
19 IF(A$="Q")GOTO27
20 GOTO16
21 CLOSE5:PRINT"(H(CPET TO HP3000 TERMINAL OP
   ERATING SYSTEM"
22 POKE59468,14:POKE167,0:D$="EAR":OPEN 5,4,1
   5,D$
23 GOSUB3
24 CLOSE5:PRINT"(H(CUPLOADING..."
25 POKE167,1:OPEN5,4,15,"EPC999997":CMD5
26 LIST
27 CLOSE 5
28 RETURN
29 REM
30 REM MACHINE CODE TO DISCONNECT
31 REM THE BASIC PROGRAMMER'S TOOLKIT
32 REM
33 PRINT"(H(C"
34 PRINT"TOOLKIT DISCONNECT ROUTINE"
35 N=832:FORI=57647TO57656
36 X=PEEK(I):POKEN,X:N=N+1:NEXTI
37 FORI=826TO831:READX:POKEI,X:NEXTI
38 FORI=842TO848:READX:POKEI,X:NEXTI
39 DATA165,119,72,165,120,72
40 DATA104,133,120,104,133,119,96
41 POKE833,24:SYS826
42 PRINT "TOOLKIT DISCONNECTED":FORI=1TO 1000
    :NEXTI
43 RETURN
```

©

Use your Atari's sound generation capabilities to automatically dial a Touch Tone telephone – it's all done with software.

Computer Controlled Telephone Dialing

Mark Savarese
Livermore, CA

After playing a number of games on my Atari computer, I decided that writing a game with the quality of commercially available products might be pretty time consuming. So, I decided to do what I like best, tinkering.

I wanted a project that would be short but interesting and needed no hardware additions to my basic system. What uses computer-like sounds besides computer music? Why, telephone beeps, of course!

After a little research, I found that Touch Tone phone systems can be dialed by providing pairs of audio tones to the mouthpiece.

	1	2	3	4	Column #
Frequency	**1209**	**1336**	**1447**	**1663**	**Hz.**
697	1	2	3	A	1
770	4	5	6	B	2
852	7	8	9	C	3
941	*	0	#	D	4
Hz.					Row #

Notice the fourth column (A,B,C,D). These buttons are not present on regular telephones, but tone pairs have been defined for them.

Suppose you wish to "dial" an 8. Just send two tones (at the same time) to the mouthpiece. An 8 would require one tone at 852 Hz and another tone at 1336 Hz.

Controlling Frequency

The Atari makes this a fairly simple task: simply send one tone on one voice and the other tone on a second voice. There is a complication, however. With normal tone generation, it is difficult to reproduce the exact frequencies needed. Never fear. The Atari provides a special mode to allow more precisely controlled frequency outputs. Two voices can be joined together to yield one "Double Precision Voice."

Line 370 connects the Atari's four voices and runs them as fast as possible to give a more precisely controlled output.

```
340 REM CONNECT REGISTERS 1 AND 2 INTO
      ONE 16-BIT REGISTER.
350 REM CONNECT REGISTERS 3 AND 4 INTO
      ONE 16-BIT REGISTER.
360 REM CLOCK ALL FOUR REGISTERS AT
      1.789790 MHZ (EXACTLY).
370 POKE 53768,120:REM SEE HARDWARE
      MANUAL.
```

The needed control rate can be calculated with the following formula:

$$\text{Control Rate} = \frac{\text{Input Frequency} = 1.78979\ \text{MHz}}{2 * \text{Output Frequency Desired}} - 7$$

The control rate is used to produce a specific output frequency. The seven in the above formula is a fudge factor used when two sound registers are connected together.

In order to get the frequencies needed to dial an 8, calculate:

$$\text{Row Control Rate} = \frac{1789790}{2 * 852} - 7 = 1043.3462 = 1043$$

$$\text{Column Control Rate} = \frac{1789790}{2 * 1336} - 7 = 662.8316 = 663$$

Notice that the numbers were rounded to the nearest whole number. This rounding results in an error of less than 0.1 per cent.

Rather than have the Atari calculate two control rates for each digit to be dialed, I pre-calculated the rates needed for the eight possible frequencies and put these values into the array T.

While this technique requires a little more typing, it saves a significant amount of CPU time. (It is rather wasteful to recalculate the same eight values repeatedly. And an array of the eight frequencies would be required anyway.)

```
170 DIM T(15,2):REM ETC.
```

The above dimension statement yields 16 sets of three values, which cover the 16 possible buttons on the telephone. Each phone button requires four bytes of information, but only three values need to be stored in the array because the fourth value is always a two by coincidence.

Here is an explanation of the program:

Lines	
0-140	Main program which calls two subroutines.
150-330	Load pre-calculated frequency control values into the array T.
340-400	Set up audio registers.
420-460	Get next "digit" to be dialed.
470-480	Convert "digit" to an ATASCII code number.
490-580	Convert "digit" to a number between 0 and 15 if it is legal, and call the tone output routine.
590	Return for a new user input if an illegal "digit" was entered.
600-700	Load frequency control values into the audio registers and output a tone pair.

These are the steps you take to use the program:

— Type the program in and SAVE it.

— Type RUN.

— The program will prompt you to enter a phone number.

— Type in the phone number you want (I suggest you use your own number to test).

— Make sure your TV volume is set moderately.

— Take your phone off the hook and check for the dial tone.

— Hold (or prop up) the mouthpiece very near your TV speaker.

— Finally, depress the return key on your computer.

— If nothing happens, you may need to adjust the TV volume up or move the phone closer to the TV speaker.

— Remember, only legal digits may be entered, no (,),- characters are allowed.

```
0 REM PHONE DIALING PROGRAM.
10 REM SET UP TONE TABLE AND SOUND RE
   GISTERS.
20 GOSUB 150
30 REM ACCEPT AN INPUT FROM THE KEYBO
   ARD.
40 PRINT
50 PRINT "ENTER PHONE NUMBER (UP TO 1
   0 BUTTONS)"
60 PRINT "FROM 0,1,2,3,4,5,6,7,8,9,A,
   B,C,D,*,#"
70 PRINT "OR A CONTROL C TO EXIT THE
   PROGRAM.":PRINT
80 INPUT NO$
90 REM CONVERT CHARACTER TO APPROPRIA
   TE TONE NUMBER AND OUTPUT TONE.
100 GOSUB 430
110 REM IF LAST CHARACTER WAS NOT A C
    ONTROL C, GO BACK FOR THE NEXT CH
    ARACTER.
120 IF ASC(CH$)<>3 THEN GOTO 40
130 SOUND 0,0,0,0:REM RESET SOUND.
140 END
150 REM SET UP THE TONE TABLE.
160 SOUND 0,0,0,0:REM INITIALIZE SOUN
    D REGISTERS,(REQUIRED).
170 DIM T(15,2):DIM CH$(1):DIM NO$(10
    )
180 T(0,0)=151:T(0,1)=3:T(0,2)=176
190 T(1,0)=221:T(1,1)=4:T(1,2)=253
200 T(2,0)=151:T(2,1)=4:T(2,2)=253
210 T(3,0)=87:T(3,1)=4:T(3,2)=253
220 T(4,0)=221:T(4,1)=4:T(4,2)=131
230 T(5,0)=151:T(5,1)=4:T(5,2)=131
240 T(6,0)=87:T(6,1)=4:T(6,2)=131
250 T(7,0)=221:T(7,1)=4:T(7,2)=19
260 T(8,0)=151:T(8,1)=4:T(8,2)=19
270 T(9,0)=87:T(9,1)=4:T(9,2)=19
280 T(10,0)=19:T(10,1)=4:T(10,2)=253
290 T(11,0)=19:T(11,1)=4:T(11,2)=131
300 T(12,0)=19:T(12,1)=4:T(12,2)=19
310 T(13,0)=19:T(13,1)=3:T(13,2)=176
320 T(14,0)=221:T(14,1)=3:T(14,2)=176
330 T(15,0)=87:T(15,1)=3:T(15,2)=176
340 REM CONNECT REGISTERS 1 AND 2.
350 REM CONNECT REGISTERS 3 AND 4.
360 REM CLOCK ALL 4 REGISTERS AT 1.78
    9790 MHZ.
370 POKE 53768,120
380 REM SET ALL VOLUMES TO ZERO.
390 POKE 53761,160:POKE 53763,160
400 POKE 53765,160:POKE 53767,160
410 RETURN
420 REM CHECK FOR AN EMPTY STRING.
430 IF LEN(NO$)<=0 THEN RETURN
440 REM STRIP OFF LEFTMOST CHARACTER
    FROM THE STRING.
450 CH$=NO$(1,1):IF LEN(NO$)=1 THEN N
    O$=""
460 IF LEN(NO$)<>0 THEN NO$=NO$(2)
470 REM CONVERT CHARACTER TO EQUIVALE
    NT ATASCII CODE NUMBER.
480 CH=ASC(CH$)
490 REM CONVERT TO A NUMBER BETWEEN 0
     AND 15.
500 REM ADJUST IF 0 TO 9.
510 IF CH<=57 AND CH>=48 THEN TN=CH-4
    8:GOSUB 610:GOTO 430
520 REM ADJUST IF A TO D.
530 IF CH<=68 AND CH>=65 THEN TN=CH-5
    5:GOSUB 610:GOTO 430
540 REM ADJUST IF a TO d.
550 IF CH<=100 AND CH>=97 THEN TN=CH-
    87:GOSUB 610:GOTO 430
560 REM ADJUST IF # OR *.
570 IF CH$="#" THEN TN=15:GOSUB 610:G
    OTO 430
580 IF CH$="*" THEN TN=14:GOSUB 610:G
    OTO 430
590 RETURN :REM RETURN IF ILLEGAL CHA
    RACTER.
600 REM PUT TONE VALUES INTO SOUND RE
    GISTERS.
610 POKE 53766,2:POKE 53764,T(TN,0):P
    OKE 53762,T(TN,1):POKE 53760,T(TN
    ,2)
620 REM TURN UP THE VOLUME ON REGISTE
    RS 2 AND 4.
630 POKE 53767,168:POKE 53763,168
640 REM WAIT A SHORT TIME.
650 FOR I=1 TO 50:NEXT I
660 REM TURN OFF THE VOLUME ON REGIST
    ERS 2 AND 4.
670 POKE 53767,160:POKE 53763,160
680 REM WAIT A SHORT TIME.
690 FOR I=1 TO 10:NEXT I
700 RETURN
```

©

ELCOMP

Books + Software for ATARI VIC-20 OSI SINCLAIR TIMEX

Payment: check, money order, VISA, MASTER-CHARGE, Euroscheck.
Orders from outside USA: add 15% shipping. CA residents add 6.5% tax
*ATARI is a registered trademark of ATARI Inc.
*VIC-20 is a registered trademark of Commodore

ELCOMP Publishing Inc.,
53 Redrock Lane
Pomona CA 91766,
Phone: (714) 623-8314

PRODUCTS FOR ATARI* 400/800 FROM ELCOMP

BOOKS:

ATARI BASIC – Learning by using
An excellent book for the beginner. Many short programs and learning exercises. All important features of the ATARI computers are described (screen drawings, special sounds, keys, paddles, joysticks, specialized screen routines, graphics, sound applications, peeks, pokes, and special stuff). Also suggestions are made that challenge you to change and write program routines.
Order # 164 — $7.95

GAMES for the ATARI Computer
This book describes advanced programming techniques like player-missile-graphics and use of the hardware-registers. Contains many ready to run programs in BASIC and one called GUNFIGHT in machine-Language.
Order # 162 — $7.95

SOFTWARE IN BASIC FOR ATARI

Invoice Writing for Small Business
This program makes writing invoices easy. Store your products in DATA statements with order-number, description, and price. The program later retrives the description and price matching to the entered order-number. The shipping cost and the discount may be calculated automatically depending on the quantity ordered or entered manually. The description to the program tells you how to change the program and adapt it to your own needs. Comes with a couple of invoice forms to write your first invoices on to it.

Order # 7201	cassette version	$29.95
Order # 7200	disk version	$39.95

Mailing List
This menu driven program allows the small business man to keep track of vendors and customers. You can search for a name or address of a certain town or for an address with a certain note. 50 addresses are put into one file.

Order # 7212	cassette version	$19.95
Order # 7213	disk version	$24.95

Inventory Control
This program is menu driven. It gives you the following options: read/store data, define items, entry editing, inventory maintenance (incoming-outgoing), reports. The products are stored with inventory number, manufacturer, reorder level, present level, code number, description.

Order # 7214	cassette version	$19.95
Order # 7215	disk version	$24.95

Programs from Book # 164
The programs from book no. 164 on cassette. (Book included)
Order # 7100 — $29.00

Game Package
Games on cassette. (Bomber, tennis, smart, cannon fodder, etc.)
Order # 7216 — $9.95

Microcomputer Hardware Handbook (845 pages)
Descriptions, pinouts and specifications of the most popular microprocessors and support chips.
A MUST for the hardware buff.
Order-No. 29
$14.95

Care and Feeding of the Commodore PET
Eight chapters exploring PET hardware. Includes repair and interfacing information. Programming tricks and schematics.
Order # 150 — $9.95

SOFTWARE IN MACHINE LANGUAGE for ATARI

ATMONA-1
This is a machine language monitor that provides you with the most important commands for programming in machine-language. Disassemble, dump (hex and ASCII), change memory location, block transfer, fill memory block, save and load machine-language programs, start programs. Printer option via three different interfaces.

Order # 7022	cassette version	$19.95
Order # 7023	disk version	$24.95
Order # 7024	cartridge version	$59,00

ATMONA-2
This is a tracer (debugger) that lets you explore the ATARI RAM/ROM area. You can stop at previously selected address, opcode, or operand. Also very valuable in understanding the microprocessor. At each stop, all registers of the CPU may be changed. Includes ATMONA-1.

Order # 7049	cassette version	$49.95
Order # 7050	disk version	$54.00

ATMAS
Macro-Assembler for ATARI-800/48k. One of the most powerful editor assemblers on the market. Versatile editor with scrolling. Up to 17k of source-Code. Very fast, translates 5k source-code in about 5 seconds. Source code can be saved on disk or cassette. (Includes ATMONA-1)

Order # 7099	disk version	$89.00
Order # 7999	cartridge version	$129.00

ATAS
Same as ATMAS but without macro-capability. Cassette-based.

Order # 7098	32k RAM	$49.95
Order # 7998	48k RAM	$49.95

ATEXT-1
This wordprocessor is an excellent buy for your money. It features screen oriented editing, scrolling, string search (even nested), left and right margin justification. Over 30 commands. Text can be saved on disk or cassette.

Order # 7210	cassette version	$29.95
Order # 7216	disk version	$34.95
Order # 7217	cartridge version	$69.00

GUNFIGHT
This game (8k machine-language) needs two joysticks. Animation and sound. Two cowboys fight against each other. Comes on a bootable cassette.
Order # 7207 — $19.95

ELCOMP FORTH for the ATARI

ELCOMP FORTH is an extended Fig-Forth-version, Editor and I/O package included. Utility package includes decompiler, sector copy, Hexdump (ASCII), ATARI Filehandling, total graphic and sound, joystickprogram and player missile. Extremely powerful!
Order # 7055 — disk — $39.95

Floating point package for ELCOMP FORTH with trigonometric functions (0 – 90°).
Order # 7230 — disk — $29.95

Learn-FORTH from ELCOMP
A subset of Fig-Forth for the beginner. On disk (32k RAM) or on cassette (16k RAM).
Order # 7053 — $19.95

Expansion boards for the APPLE II

The Custom Apple + Other Mysteries
A complete guide to customizing the Apple Software und Hardware
Order-No. 680 — $24.95

We also stock the boards which are used in the book "The Custom Apple ..." (bareboards)

6522 I/O Board No. 605	$39.00
EPROM Burner No. 607	$49.00
8K EPROM/RAM Board No. 609	$29.00
Prototyping board for the Apple II No. 604	$29.00
Slot repeater board for the Apple II No. 606	$49.00

Order two boards and get the book free!

No. 6153 Learn FORTH for the Apple II	(C)	$19.95
No. 6154 Learn FORTH for the Apple II	(D)	$24.95
No. 6155 ELCOMP FORTH for the Apple II (Extended Fig-Forth, Editor, graphics package)	(D)	$39.95
No. 6156 Floating point for ELCOMP FORTH	(D)	$29.95

Hardware – ADD-ONS for ATARI

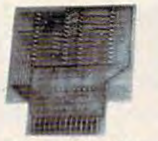
PRINTER INTERFACE
This construction article comes with printed circuit board and software. You can use the EPSON printer without the ATARI printer interface. (Works with gameports 3 and 4).
Order # 7211 — $19.95

RS-232 Interface for your ATARI 400/800
Software with connector and construction article.
Order # 7291 — $19.95

EPROM BURNER for ATARI 400/800
Works with gameports. No additional power supply needed. Comes compl. assembled with software (2716, 2732, 2532).
Order # 7042 — $179.00

EPROM BURNER for ATARI 400/800 KIT
Printed circuit board incl. Software and extensive construction article.
Order #7292 — $49.00

EPROM BOARD (CARTRIDGE)
Holds two 4k EPROMs (2532). EPROMs not included.
Order # 7043 — $29.95



EPROM BOARD KIT
Same as above but bare board only with description.
Order # 7224 — $14.95

ATARI, VIC-20, Sinclair, Timex and OSI

New – for your ATARI 400/800

Astrology and Biorythm for ATARI (cass. or disk).
Order # 7223 — $29.95

Birth control with the ATARI (Knaus Ogino)
Order # 7222 — cass. or disk — $29.95

Books + Software for VIC-20 (requires 3k RAM Exp.)

# 4870 Wordprocessor for VIC-20, 8k RAM	$19.95
# 4883 Mailing List for VIC-20, 16k RAM	$14.95
# 141 Tricks for VICs - The VICstory Progr.	$9.95
# 4880 TIC TAC VIC	$9.95
# 4881 GAMEPACK I (3 Games)	$14.95
# 4885 Dual Joystick Instruction	$9.95

INPUT/OUTPUT Programming with your VIC
Order # 4886 — $9.95

# 4896 Miniassembler for VIC-20	$19.95
# 4881 Tennis, Squash, Break	$9.95
# 4894 Runfill for VIC	$9.95

Universal Experimenter Board for the VIC-20
(Save money with this great board). This board plugs right into the expansion slot of the VIC-20. The board contains a large prototyping area for your own circuit design and expansion. The construction article shows you how to build your own 3k RAM expander and ROM-board.
Order # 4844 — $18.95

Software for SINCLAIR ZX-81 and TIMEX 1000

# 2399 Machine Language Monitor	$9.95
# 2398 Mailing List	$19.95

Programming in BASIC and machine language with the ZX-81 (82) or TIMEX 1000.
Order # 140 — (book) — $9.95

Books for OSI

# 157 The First Book of Ohio	$7.95
# 158 The Second Book of Ohio	$7.95
# 159 The Third Book of Ohio	$7.95
# 160 The Fourth Book of Ohio	$7.95
# 161 The Fifth Book of Ohio	$7.95

# 151 8K Microsoft BASIC Ref. Man.	$9.95
# 152 Expansion Handbook for 6502 and 6802	$9.95
# 153 Microcomputer Appl. Notes	$9.95

Complex Sound Generation
New revised applications manual for the Texas Instruments SN 76477 Complex Sound Generator.
Order # 154 — $6.95

Small Business Programs Order # 156
Complete listings for the business user. Inventory, Invoice Writing, Mailing List and much more. Introduction to Business Applications. — $14.90

Does your VIC need to communicate with a megamemory mainframe? Or use a modem for other specialized data transfers? This flexible program can handle a variety of communications needs for the VIC.

From VIC-20 To Mainframe

Ulrich Merten
Pittsburgh

We recently wanted to use our VIC-20 to communicate with a mainframe computer running under IBM's VM/CMS interactive system. We wanted to access our files on the mainframe, to write into those files using the system editor, and to list BASIC programs developed on the VIC-20 to the larger system. The system has a 300 baud ASCII port which is accessible by telephone.

For this purpose we used a Bizcomp Versamodem and rewrote the software to fit our situation. In the process, we had to learn to convert a BASIC program listing into a cassette data file on the VIC-20. The methods we used should be of interest to other VIC-20 users with similar communications challenges.

The terminal program we wrote begins by opening a file directed to the VIC-20's RS-232 interface, specifying a "file name" which selects a communications speed of 300 baud. The program has two major segments which work alternately, one looking for input from the VIC keyboard, using the "GETB$" command at line 160, and the other looking for input from the mainframe, using the "GET#10,C$" command at line 560. The bulk of the program is concerned with converting the VIC characters to ASCII, and also with translating the ASCII code arriving from the mainframe into the corresponding VIC characters on the screen.

Because we were interested in working with text in the system editor, we set the VIC keyboard in the mode in which it types upper- and lowercase letters to the screen. At lines 305 and 330, the VIC codes for the lower- and uppercase alphabetic characters, respectively, are incremented to give the appropriate ASCII codes for the same letters. The remaining lines from 300 through 335 take care of a variety of problems, two of which deserve mention here.

Line 300 calls a subroutine which sends a quotation mark ahead of the ASCII code for "#", because we found that without this provision, the # symbol was not transmitted. This was the only case we encountered where the quotation mark was necessary, but there may be others. Line 301 sets the F1 key on the VIC for a purpose we'll discuss below. Obviously, additional lines could be written into the program at this point to accommodate other function keys. Line 400 causes our input to be printed on the screen so that we can see what we're doing, and line 500 sends it on to the RS-232 interface and the mainframe.

The codes coming back from the host are pure ASCII and have to be converted to what the VIC uses in this screen mode, and that's what lines 530-610 are all about. Lines 555 and 600 take care of upper- and lowercase alphabetic characters, respectively, and line 540 translates the signal sent out by the mainframe at the end of each line. The rest of the lines in the range 530-610 are for housekeeping.

The program increments most of the ASCII codes which do not represent alphanumeric characters by 160, causing them to print out as VIC graphic characters. This has the advantage that if and when the mainframe sends back one of these characters, you see it on the screen and can identify it. The feature can be eliminated as an unnecessary nuisance once you know what is being sent your way!

Buffer Relief

When we tried to "LIST" to the RS-232 interface using an early version of this program, we found that the buffer quickly overloaded and that our transfer attempts were unsuccessful. So we converted the program we wanted to list into a data file on our cassette, using the command series:

```
CLOSE1:OPEN1,1,1:CMD1:LIST
```

This proved a successful stratagem, except that when we used the "GET" command to read this file, we didn't get the last few program lines. We solved this problem by adding a few lines of "pound signs," CHR$(92), at the end of the program and sacrificing those.

Lines 190-225 of the terminal program exist to take care of these program listings. If the F1 key is depressed, the variable "z" is set equal to one, and the next time the program passes through line 150 it opens the cassette file at line 190, and starts reading the contents. Lines 215 and 220 tell the

program what to do with some special VIC characters we had in the programs, which ASCII can't handle, and line 230 makes the program print out in uppercase at the mainframe. Line 210 watches out for those "pound signs" we added at the end of the listed program, and halts the proceedings when it sees the first one.

Our first attempts to run this version of the program were unsuccessful, because data kept going out at the end of each line, before the mainframe was ready for them. That's why we added the "IF" statement to line 510. The mainframe sends out a period when it's ready for a new line, and now the VIC waits for that period to come back before proceeding, after each time it sends out a carriage return.

We've found it possible to work effectively between the VIC-20 and the mainframe using this program, and are very pleased with the ease with which we can modify the "IF" statements to meet various demands such as printing substitute characters for those not available in ASCII.

```
100 OPEN1Ø,2,3,CHR$(38)+CHR$(16Ø)
15Ø IFZ=1THEN19Ø
155 IFZ=2THEN2ØØ
16Ø GETB$
165 IFB$=''THEN51Ø
17Ø X=ASC(B$)
18Ø GOTO3ØØ
19Ø CLOSE1:OPEN1,1,Ø:Z=2:GET#1,B$:GET#1,B$
2ØØ GET#1,B$
2Ø5 X=ASC(B$)
21Ø IFX=92THENZ=Ø:CLOSE1:GOTO51Ø
215 IFX=18ØRX=146THEN51Ø
22Ø IFX=147THENX=99:GOTO4ØØ
225 IFX=34THENX=39:GOTO4ØØ
23Ø IFX>64ANDX<91THEN4ØØ
3ØØ IFX=35THENGOSUB645:GOTO4ØØ
3Ø5 IFX>64ANDX<91THENX=X+32:GOTO4ØØ
31Ø IFX>127ANDX<133THENX=32:GOTO4ØØ
315 IFX=133THENZ=1:GOTO15Ø
325 IFX>133ANDX<192THENX=32:GOTO4ØØ
33Ø IFX>192ANDX<224THENX=X-128:GOTO4ØØ
335 IFX>224THENX=32:GOTO4ØØ
4ØØ PRINTB$;
5ØØ PRINT#1Ø,CHR$(X);
51Ø GET#1Ø,C$:IFX=13ANDZ<>ØANDC$<>'.'THEN51Ø
52Ø IFC$=''THEN62Ø
525 Y=ASC(C$)
53Ø IFY=13THEN62Ø
54Ø IFY<32THENY=Y+16Ø:GOTO615
555 IFY>64ANDY<91THENY=Y+128:GOTO615
56Ø IFY=96THENY=32:GOTO615
6ØØ IFY>96FANDY<123THENY=Y-32:GOTO615
6Ø5 IFY>122ANDY<127THENY=Y-64: GOTO615
61Ø IFY=127THEN63Ø
615 C$=CHR$(Y)
62Ø PRINTC$;
63Ø IFST=ØTHEN15Ø
64Ø PRINT'ERROR'
645 PRINT#1Ø,CHR$(34);
646 RETURN
65Ø END
```

©

VIC-20* SOFTWARE

GAMES

Cartridge:

Spiders of Mars	$40.00
Outworld	40.00
Cloud Burst	32.00
Alien Blitz	32.00
Renaissance	40.00
Amok	32.00
Meteor Run	40.00
Sats & Mets	40.00
Skibbereen	32.00

Cassette:

Simon	$12.00
Amok	20.00
Subchase	20.00
Alien Blitz	20.00
The Alien	20.00
3-D Maze	12.00
Raceway	12.00
Kosmic Kamikaze	20.00
Super Driver	15.00 **

*VIC-20 & Commodore-64 are registered trademarks of Commodore Business Machines.

**Prices and Availability Subject to Change Without Notice.

Business:

ViCalc	$12.00
ViCat	20.00
ViCheck	20.00
ViTerm A	16.00

Educational:

Skymath	$20.00
Space Div.	12.00
Super Hangman	16.00

Hardware:

Buti-3K ROM	$72.00
Spokesman	150.00

Voice Synthesizer for (Commodore-64*), VIC-20* **

ALL ORDERS ADD $2.00 SHIPPING
California Residents Add 6% Sales Tax
Dealers Please Inquire

TO ORDER SEND CHECK OR MONEY ORDER TO:
MOOSEWARE INCORPORATED
Post Office Box 17868, Irvine, California 92713

WRITE FOR FREE BROCHURE & ASK ABOUT THE MOOSE CLUB.
CLUB MEMBERS RECEIVE SPECIAL DISCOUNTS ON ALL PRODUCTS SOLD.

VISA

NEW VIC SOFTWARE VIC

MasterCard

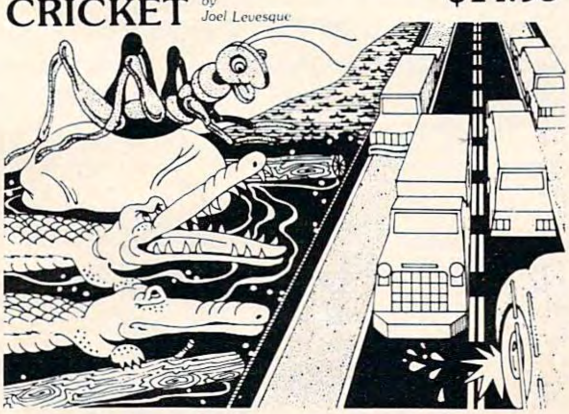
COMPUTERMAT • BOX 1664, DEPT C
LAKE HAVASU CITY, ARIZONA 86403

(602) 855-3357

WRITE FOR FREE CATALOG OF VIC SOFTWARE

CRICKET
by Joel Levesque

$14.95

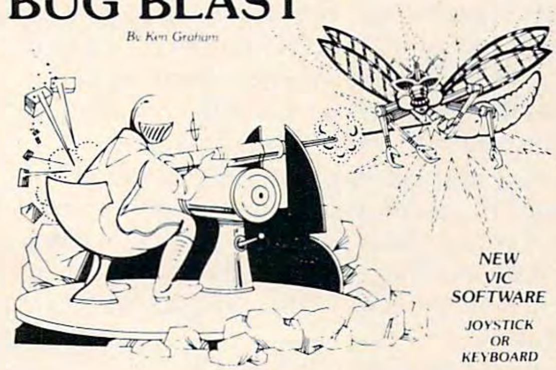


WE WELCOME YOUR PERSONAL CHECK

PLEASE ADD $1.00 PER ORDER FOR SHIPPING

BUG BLAST
By Ken Graham

$14.95



ALL VIC SOFTWARE RUNS IN STANDARD VIC.

Two programs in Applesoft to lend variety to your program menus.

Apple Menu

Robert J. Beck
Minneapolis, MN

An often-ignored but essential component of many programs is the menu, or list of options. Here are a couple of variations that may make your programs a bit more interesting. Each of these imaginary menus consists of four choices: "First," "Second," "Third," and "Quit." The first three choices return you to the menu after printing a word; the "Quit" option stops the program.

An Alphabet Menu

```
10 R$ = "FSTQ"
20 PRINT "FIRST, SECOND, THIRD, OR QUIT"
30 PRINT "F,S,T, OR Q?";
40 GET Z$: PRINT
50 FOR I = 1 TO 4
60 IF Z$ = MID$(R$,I,1)<>THEN ON I GOTO 500,6
   00,700,800
70 NEXT I
80 PRINT "PLEASE CHOOSE";
90 GOTO 30
500 PRINT "FIRST": GOTO 30
600 PRINT "SECOND": GOTO 30
700 PRINT "THIRD": GOTO 30
800 PRINT "QUIT": END
```

In the example above, you make a choice by typing a letter. The letter is an abbreviation of the choice. Abbreviations don't use much space; you can use a one or two-line menu, thus preserving previous screen output.

Line 10 sets up a string variable that is a concatenation of the abbreviations. Matching the input from line 40 with this string generates an index value that is used in the ON GOTO of line 60. Essentially, R$ is used as a table, and lines 50-70 perform a table lookup. An array, such as R$(1) - R$(4), could substitute for R$, but that's a waste of memory. Note that, especially with long option lists, this method is superior to using a series of IF/THEN statements to make the branch.

You type nothing in when using the arrow menu. Instead, you move an arrow until it points at the desired choice, then you press the RETURN key. The only way that you can accidentally make an unwanted choice is by being too hasty with the RETURN key.

Let's take it one line at a time. Line 10 initializes some important variables: HT (horizontal tab) is the number of spaces that the option list is tabbed over, VT (vertical tab) is the vertical line number at which the list begins, N is the number of options,

An Arrow Menu

```
10 HT = 10: VT = 7: N = 4: T = VT
20 HOME: PRINT: "TEST MENU"
30 VTAB VT
40 FOR I = 1 TO N
50 READ CHOICE$: HTAB HT: PRINT CHOICE$: PRIN
   T
60 NEXT
70 DATA FIRST,SECOND,THIRD,QUIT
80 VTAB22: PRINT "TYPE 'D' TO MOVE DOWN, 'U'
   TO MOVE UP."
90 PRINT "HIT RETURN TO SELECT."
100 POKE 33,3: POKE 32,HT - 5: VTAB VT
110 HTAB 1: PRINT "->" ;: GET C$
120 IF C$ = "D" AND T < N + VT - 1 THEN HTAB 1
    :PRINT" ":PRINT:T = T + 1:GOTO110
130 IF C$ = "U" AND T>VT THEN HTAB 1:CALL-868:
    VTAB PEEK(37)-1:T = T - 1:GOTO110
140 IF C$ = CHR$ (13) THEN TEXT: ON T - VT + 1
     GOTO 500,600,700,800
150 GOTO 110
500 HOME: SPEED = 50: PRINT"FIRST": SPEED = 25
    5: GOTO 10
600 HOME: SPEED = 50: PRINT"SECOND": SPEED = 2
    55: GOTO 10
700 HOME: SPEED = 50: PRINT"THIRD": SPEED = 25
    5: GOTO 10
800 HOME: PRINT "QUIT": END
```

and T is used to keep track of which choice the arrow points at. Line 20 prints the title, line 30 tabs to the preset vertical line, and lines 40-80 print the menu. The first POKE in line 90 sets line width to three spaces; the second POKE sets the left margin five spaces to the left of the menu. A VTAB to the top of the menu list completes the preparations for printing the arrow and GETting a keypress at line 110.

If T equals N + VT - 1, the arrow is at the bottom of the list; if T equals VT, then the arrow is at the top. Lines 120 and 130 illustrate two slightly different ways of moving the arrow. Line 120 prints blank spaces over it, while line 130 uses a monitor subroutine to erase it from the screen. Note that, though the cursor is moved two lines upward, the VTAB in line 130 is for PEEK (37) - 1. This is because VTAB numbers the screen lines from 1 to 24, but PEEK (37) uses 0 to 23. Unless your program uses the same line width and margin as the menu, you'll need the TEXT in line 140 to reset the text window. ©

With this program it's much easier to generate sprites on the Commodore 64. You can draw a shape, examine it, modify it, and then save it in DATA statements ready to use anywhere.

A Shape Generator For The Commodore 64

Donald A. Pitts
Houston

The sprite graphics of the Commodore 64 is an attractive feature. It allows you to create rapid movement of complex shapes with one shape disappearing behind another in an apparent three dimensional display. The sprite is a 24 by 21 grid with each grid element being one bit. Three bytes are positioned side by side to make up the 24 bits. To manually build a sprite, a user would have to construct the grid on graph paper, draw the desired figure, and then determine the value of each byte according to the bits which are enabled (see Figure 1). Although Commodore set up the sprite system in a way that is very logical, the shape generator program makes the job of generating sprites easier and much faster.

The program, Shape Gen, allows you to draw a shape on a 24 by 21 grid on the screen, preview what the sprite will look like, modify the sprite further, and finally save the sprite information in BASIC DATA statements. To the left of the grid is a menu of available commands. Below that is a space for the sprite to be displayed, in both normal and enlarged sizes. This allows the user to determine the exact way the sprite will be displayed prior to saving it. When the shape is deemed perfect, pressing the "←" key will erase the Shape Gen program and leave behind the data statements that describe the sprite you have just drawn.

Drawing The Shape

Four keys (I = up, J = left, K = right, M = down) are used to move the Shape Gen cursor. Either shift key may be pressed to draw while the cursor is either moving or stationary. The Commodore key works the same way, except its function is to delete pixels on the grid. When you have finished drawing the shape or want to see what the shape would look like as a sprite, press F1 (located in the upper right of the keyboard).

The program will tell you it is compiling the shape at this point. In a few moments the cursor will reappear, and two shapes will appear at the left side of the screen. At this point you may either change the shape or transform the shape into DATA statements. Should you desire to start over, you may depress the CLR/HOME key.

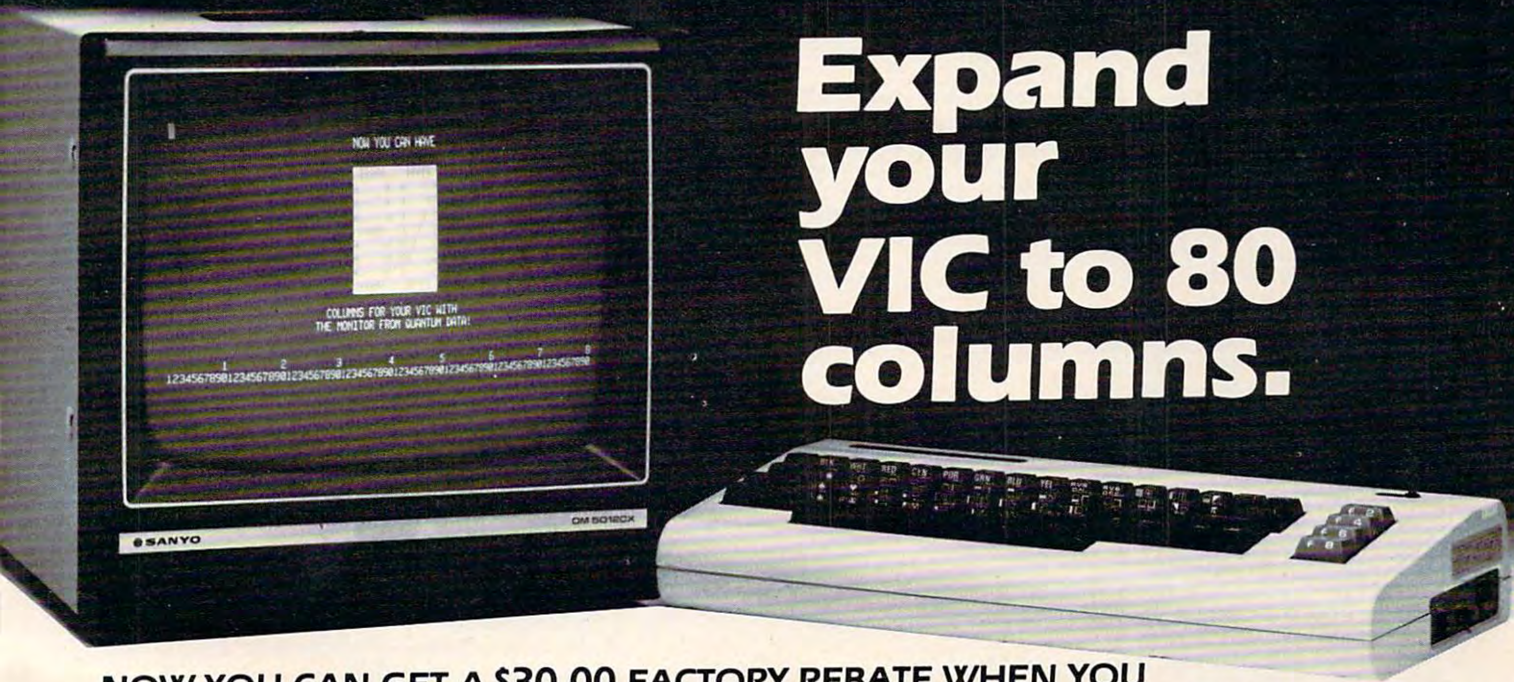
Once you have compiled the shape for the last time, press "←"; a warning will appear because this option erases the Shape Gen program leaving behind the DATA statements with the data necessary to re-create the sprite in other programs. Press "Y" or "N" in response to "continue?". If you respond "Y" the DATA statements will be listed to the screen, and Shape Gen will end execution. Now you are free to add your own program to the DATA statements to manipulate the shapes on the screen. To do this, it will be helpful to read section six of the *Commodore 64 User's Guide*.

The following is an example of a group of DATA statements generated by the Shape Gen program together with a BASIC program that will move this sprite from the upper middle to the lower left of the screen.

```
1 REMAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
   AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
2 REMAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
   AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
3 REMAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
   AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
4 REMAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
   AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
50 REM
60 REM
100 PRINTCHR$(147):FORX=55296TO56295:POKEX,3:N
    EXT
105 PRINTCHR$(145);TAB(13);CHR$(18);"SHAPE GEN
     1.0";CHR$(146)
107 PRINT:PRINT" ";CHR$(18);"I";CHR$(146);"
    = UP"
109 PRINT" ";CHR$(18);"M";CHR$(146);"    = DOWN
    "
111 PRINT" ";CHR$(18);"J";CHR$(146);"    = LEFT
    "
113 PRINT" ";CHR$(18);"K";CHR$(146);"    = RIGH
    T"
114 PRINT" ";CHR$(18);"COM";CHR$(146);" = DELE
    TE"
115 PRINT" ";CHR$(18);"SHFT";CHR$(146);"= DRAW
    "
116 PRINT" ";CHR$(18);"_";CHR$(146);"    = DATA
    "
117 PRINT" ";CHR$(18);"F1";CHR$(146);"   = SHAP
    E"
```

22-40-80 HIKE!

Expand your VIC to 80 columns.



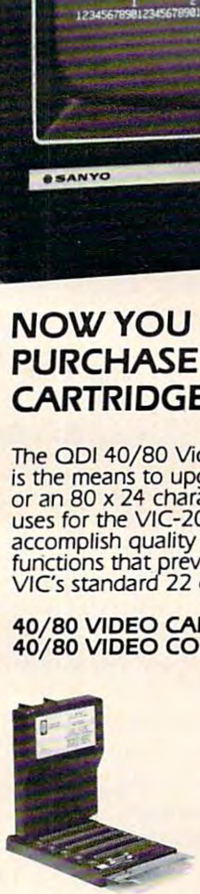
NOW YOU CAN GET A $30.00 FACTORY REBATE WHEN YOU PURCHASE A QUANTUM DATA VIDEO CARTRIDGE OR VIDEO COMBO CARTRIDGE!!

The QDI 40/80 Video Cartridge and Video Combo Cartridge is the means to upgrade the VIC-20 computer to a 40 x 24 or an 80 x 24 character display, providing a wealth of new uses for the VIC-20. With appropriate software, you can now accomplish quality word processing and various business functions that previously were very difficult with only the VIC's standard 22 character video display.

40/80 VIDEO CARTRIDGE 0-K memory: $219.95
40/80 VIDEO COMBO CARTRIDGE w/16-K RAM: $319.95

- Features a high quality 8 x 8 dot matrix.
- Character-by-character reverse video attributes allowing adjacent characters to have different attributes.
- All features are accessible through BASIC using POKE commands.
- Black & White composite video.
- Plugs directly into the VIC-20 memory expansion slot or the QDI Mini-Mother and Maxi-Mother boards.
- Contains 2K of CMOS internal video RAM; no system RAM is used by the Video Cartridge.
- 40 Columns can be viewed using your home T.V. while 80 columns require using a video monitor.

Offer expires 12:00 midnight US, pacific standard time, December 31, 1982.



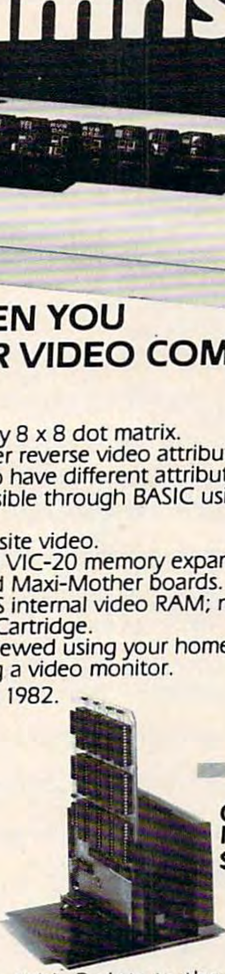
QDI Maxi-Mother: $99.95

SPECIAL INTRODUCTORY OFFER $15.00 FACTORY REBATE!!!

- Adds 6 slots to the memory expansion port.
- Three slots fixed and three slots power switch selectable.
- On board master reset button allows you to reset cold start your VIC-20 without powering down.
- .1" Cannon rear connector accepts common external power supply when fuse is removed.
- Simple plug-in installation.
- Small size 5 x 8 inches encased.

QDI RS-232C Interface: $49.95

- 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



QDI Mini-Mother: $59.95

- 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

QUANTUM DATA, INC. (714) 966-6553
DEALER HOT LINE (714) 754-1945
3001 Redhill Bldg. 4, Suite 105, Costa Mesa, CA 92626

Shipping cost not included — VISA & MASTERCARD accepted. Dealer inquiries are welcomed. For further information please contact your local Commodore dealer or QUANTUM DATA for a listing of dealers near you. All the above prices retail in US dollars.

```
118 PRINT" ";CHR$(18);"HOME";CHR$(146);"= CLEA
    R"
119 PRINT:PRINT:PRINT" SMALL":PRINT" SHAPE"
12Ø PRINT:PRINT:PRINT:PRINT" LARGE":PRINT" SHA
    PE"
121 PRINT:PRINT:PRINT:PRINT:PRINTTAB(13);CHR$(
    18);"DRAWING MODE   ";CHR$(146);
129 DIMG(62):V=53248:UL=1Ø78:FORX=ØTO25:POKEUL
    +X,42:POKEUL+22*4Ø+X,42:NEXT
13Ø FORX=ØTO22:POKEUL+X*4Ø,42:POKEUL+25+X*4Ø,4
    2:NEXT
14Ø PT=UL+41
15Ø SL=PEEK(PT):POKEPT,81:FORX=ØTO8Ø:NEXT:POKE
    PT,SL
16Ø A=PEEK(197):C=PEEK(653)
162 IFC=1THENPOKEPT,16Ø
164 IFC=2THENPOKEPT,32
17Ø IFA=33THENM=-4Ø:GOTO3ØØ
18Ø IFA=34THENM=-1:GOTO3ØØ
19Ø IFA=37THENM=1:GOTO3ØØ
2ØØ IFA=36THENM=4Ø:GOTO3ØØ
21Ø IFA=51THENPRINTCHR$(147):POKEV+21,Ø:RUN1Ø5

22Ø IFA=4THEN4ØØ
225 IFA=57THEN5ØØ
23Ø GOTO15Ø
3ØØ IFPEEK(PT+M)=42THEN15Ø
31Ø PT=PT+M:GOTO15Ø
4ØØ FORX=1TO15:PRINTCHR$(157);:NEXT:PRINTCHR$(
    18);"COMPILING SHAPE";CHR$(146);
4Ø1 N=Ø:Z=8:FORY=1TO21:FORX=1TO24:P=PEEK(UL+Y*
    4Ø+X):Z=Z-1
41Ø IFZ=-1THENC=Ø:Z=7:N=N+1
42Ø IFP=16ØTHENC=C+2^Z
425 IFZ=ØTHENPOKE832+N,C:G(N)=C
43Ø NEXT:NEXT
44Ø POKEV+21,12:POKE2Ø42,13:POKE2Ø43,13:POKEV+
    4,9Ø:POKEV+5,15Ø
445 POKEV+6,8Ø:POKEV+7,18Ø:POKEV+23,8:POKEV+29
    ,8
45Ø FORX=1TO15:PRINTCHR$(157);:NEXT
46Ø PRINTCHR$(18);"DRAWING MODE   ";CHR$(146);
    :GOTO15Ø
5ØØ DT=Ø:CU=PEEK(43)+PEEK(44)*256+4:POKEV+21,Ø

5Ø1 FORX=1TO25:PRINTCHR$(157);:NEXT
5Ø2 PRINT"DATA WILL ERASE PROGRAM -- CONTINUE?
    ";:POKE198,Ø
5Ø3 GETA$:IFA$<>"Y"ANDA$<>"N"THEN5Ø3
5Ø4 IFA$="N"THENFORX=1TO26:PRINTCHR$(157);:NEX
    T
5Ø5 IFA$="N"THENFORX=1984TO2Ø23:POKEX,32:NEXT:
    GOTO46Ø
5Ø6 PRINTCHR$(147);TAB(13);"PUTTING SHAPE INTO
    "
5Ø7 PRINTTAB(13);"DATA STATEMENTS"
51Ø POKECU,131:CN=1
53Ø D$=STR$(G(DT)):FORX=2TOLEN(D$):C=ASC(MID$(
    D$,X,1))
532 POKECU+CN,C:CN=CN+1:NEXT
535 DT=DT+1:IFDT=63THEN56Ø
54Ø IFCN>71THENFORX=CNTO75:POKECU+X,32:NEXT:PO
    KECU+76,Ø:CU=CU+81:GOTO51Ø
55Ø POKECU+CN,44:CN=CN+1:GOTO53Ø
56Ø FORX=CNTO75:POKECU+X,32:NEXT:FORX=76TO78:P
    OKECU+X,Ø:NEXT:LIST:END
```

Should you desire to save the DATA statements and merge them with other programs, you should refer to Jim Butterfield's article in the June 1982 **COMPUTE!** (p. 158) on merging VIC-20 programs. His technique will work with one addition. After you have saved the program on tape and are trying to merge it, you will be unable to clear the screen as Butterfield tells you to do. At that point hit the Commodore key and continue with the rest of the commands.

A Note On Entering The Program

1) The first four REM statements must be typed in, in order to use the data option of the program. They must be typed in with no spaces, exactly as they appear in the listing. Seventy-five A's should follow each REM.

2) When writing the program, I specifically used CHR$ statements in place of cursor control characters embedded within print statements. I hope this will aid new Commodore 64 users in typing in the program.

3) Please save the program at least once before running it for the first time since the program is designed to erase itself when certain options are exercised.

Program Description

Line no.	
1-4	REM statements that will contain the shape DATA statements. These four lines must be typed in with no blanks anywhere including between the line # and REM and also between REM and the first A. There must be 75 A's in each line.
100	Clears screen; clears color to cyan.
105	Moves cursor up one line; turns on reverse print; prints title; turns off reverse print.
107-118	Print command keys in reverse lettering with a very brief accompanying description.
119-120	Print labels at the places where the shape will be shown if compiled.
121	Prints current mode in reverse lettering.
129	Sets up array G to store shape data. V is starting memory location in the video chip; UL is upper left of shape drawing region. Draws upper and lower horizontal lines of asterisks to indicate the boundaries of the shape drawing region.
130	Draws left and right vertical line of asterisks.
140	PT is the cursor position within the drawing region.
150	SL is the character underlying the cursor; displays cursor; time delay; redisplays character.
160	Looks at keyboard; looks at status of Shift and Commodore keys.
162	Fills in area under cursor if either Shift key is pressed.
164	Erases area under cursor if Commodore key is pressed.
170	Moves up if I key is pressed.
180	Moves left if J key is pressed.
190	Moves right if K key is pressed.
200	Moves down if M key is pressed.
210	Erases screen and shapes if CLR/HOME key is pressed.
220	Compiles shape if F1 key is pressed.
225	Takes shape data and puts into DATA statements if the left arrow key is pressed. This command will erase the program, so make sure that you copy the program before using this option.
300	If area cursor is to move to is an asterisk, then do not move cursor.
310	Adds movement value to cursor pointer and reenters

main routine.

400 Moves cursor left until at beginning of mode message; changes message.

401 N is the counter for shape data; Z is the bit position within the current byte of shape data being compiled. Scans along shape drawing region, from left to right each row, starting at the top moving toward the bottom. P is the character at the present scan position. Decrements bit position. If finished compiling current byte, then sets bit position to bit 7 and increments shape data count.

410 If finished compiling current byte, then resets shape data byte to zero.

420 If character at the present scan position is a solid box, then sets current bit position to 1.

425 If current byte finished, then POKEs it into memory block 13 and also saves it in array G.

440 Activates sprites 2 and 3; sets sprite 2's data pointer to memory block 13; sets sprite 3's data pointer to block 13; sets sprite 2 to coordinates (90,150).

445 Sets sprite 3's coordinates to (80,180); expands sprite 3 in both X and Y direction.

450 Moves cursor left to start of mode message.

460 Changes mode message.

500 DT is the count of data stored in BASIC program. Erases any sprites on screen.

501 Moves cursor left to beginning of bottom line.

502 Prints warning message that only the data statements that are generated will be left.

503 Gets response in A$.

504 Moves cursor back to start of mode message.

505 Clears bottom line of screen and branches to 460.

506-507 Clear screen; print message indicating action.

510 POKEs DATA token; CU is position within BASIC line.

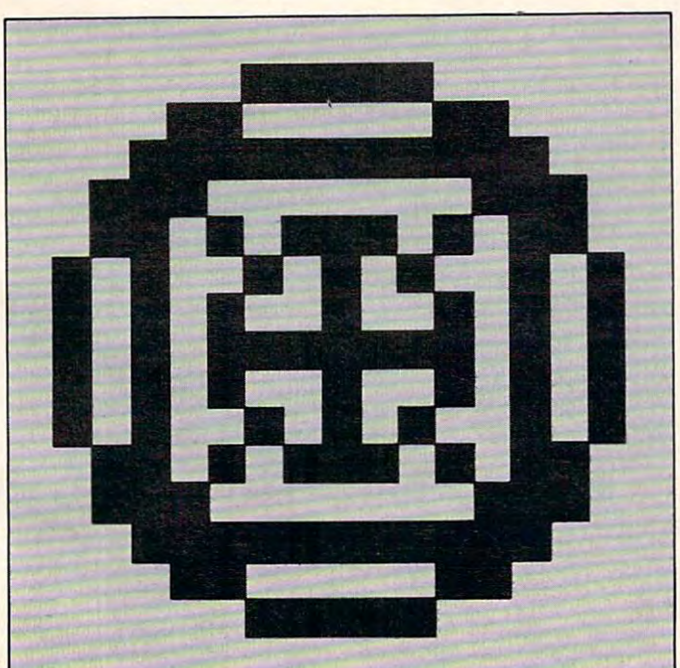
530-532 Set shape data to string; POKE string character by character into BASIC line while increasing CN.

535 Increases data count; if all data finished, then branches to 560.

540 If current BASIC line hasn't enough space for any more data, then fills remaining bytes with spaces, adds a zero to the end, and sets BASIC line pointer to next BASIC line.

550 Puts comma in line; increases pointer within line.

560 Fills remaining locations in last DATA line with spaces; adds three zeroes to end; LISTs program; ends program.



Sprite Bit Map

Sprite Byte Map

0	0	0
0	0	0
0	0	0
0	0	0
0	62	0
0	193	128
1	255	192
3	128	224
3	93	96
5	42	80
5	73	80
5	127	80
5	73	80
5	42	80
3	93	96
3	128	224
1	255	192
0	193	128
0	62	0
0	0	0
0	0	0

©

This will save any standard Atari BASIC screen display very fast. The program is designed for the Atari with cassette (16K) or disk (24K). Once saved, pictures take in six minutes (tape) or in less than 15 seconds (disk).

Atari Screen Save

Richard S. Waller, Seven Hills, OH

The superb graphics capabilities of the Atari are exploited by many programs which create beautiful screen displays. The problem is saving these pictures for quick display at a later time. My Atari Screen Save Utility is one of the fastest ways to save and recall pictures from a disk or cassette.

For example, Mike Kinnamon's "Supercube Update" in **COMPUTE!** (August 1981, #15) claims four and a half minutes to save a Graphics 7 picture to disk. Substituting my Screen Save Utility code at lines 4002 to 5008 should cut the time to under ten seconds. I've chosen a different program which uses Graphics 8 to demonstrate my utility.

The Micro Technology Unlimited advertisement on page 41 of the November 1981 **COMPUTE!** shows, on a modified PET, a hi-res graphics picture that looks like a man's hat. It really intrigued me – could a standard Atari do it? I entered the program listing into my Atari. Slowly the display emerged, but upside down.

It seems that MTU has given the X,Y coordinates their classical position of 0,0 in the lower left corner, you lucky PET people. But after I adjusted the program to the Atari coordinate system with 0,0 in the upper left corner, the program ran and produced the same hi-res picture as the ad. It took almost three and a half hours on the Atari to draw the picture. Mission accomplished, but I wanted my computer back, and three and a half hours were lost at the flick of the off switch.

Add It To Any Program

In the same issue of **COMPUTE!**, I also read Bill Wilkinson's "Insight: Atari" article on the flexibility of I/O with the Atari operating system. Obviously, saving a TV picture should be a piece of cake, so I wrote this Atari Screen Save Utility. Now the 8K hi-res display can be saved to disk in about 15 seconds. Then, by changing one variable from an 8 to a 4, the same routine will read the disk and display the saved picture (that took three and a half hours to draw) back onto the TV again only in a mere 15 seconds.

I've tried to write the program so that it can be added to any program like Supercube. It does require the display to start with a BASIC Graphics command, and it uses the first 20 bytes of page six for the machine language code to get and put multiple bytes to and from the disk or (if D: is replaced with C:) to a cassette.

If you don't want to wait three hours to test the program, just increase the STEPs in lines 60 and 100 from one to some larger number like eight. For the final three hour picture, the time can be reduced to two and a half hours by turning off the screen display. This is done with a POKE 559,0 at the start of the display code, with a POKE 559,34 at the end to turn the screen display back on. (See "Unleash The Power Of Your Atari CPU," by Ed Stewart, in **COMPUTE!**, April 1981, #11.)

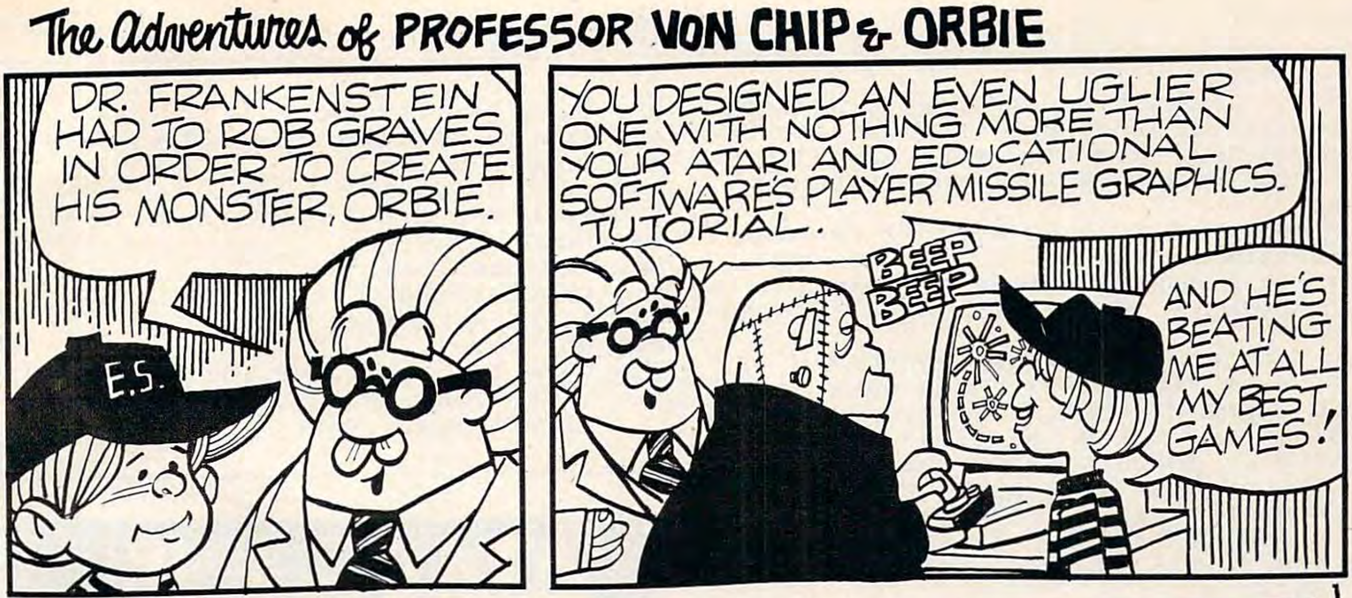
Remember, the program is designed to save the entire screen, so the instructions cannot be displayed when appropriate, but must be displayed only before the picture is drawn. The keys that you press will not display, but the computer will follow them anyway. So read the instructions carefully at the start of the program and enjoy fast recall of your TV screen displays from tape or disk.

This program will run on an Atari with 16K and cassette or 24K with DOS. Once an 8K picture is saved, it can be displayed in under six minutes from tape or under 15 seconds from disk. Other graphics modes will take much less time.

Machine Code Listing

```
    PLA        GET # OF ARGUMENTS
    CMP #1     CHECK FOR 1 ARG.
    BNE ER     RETURN IF NOT 1
    PLA        DISREGARD HI BYTE OF ARG.
    PLA        LO BYTE OF ARG.
    STA $327   PUT ARG. IN IOCB#3
    LDX #$30   INDEX TO IOCB#3
    JSR $E456  JUMP TO DO I/O
ER  STA $D5    STORE ERROR FLAG FOR BASIC
    LDA #0     ZERO OTHER BYTE FOR BASIC
    STA $D4    BECOMES RESULT OF USR CALL
    RTS        RETURN TO BASIC
```

```
0 REM SCREEN SAVE UTILITY PROGRAM BY.
  ..R.S.WALLER   12/26/81
1 DIM FN$(17)
2 IN408=8:POKE 764,255:GRAPHICS 0:? "
  {6 SPACES}SCREEN SAVE OPTION"
```



TRICKY TUTORIALS (tm)

There are many things that the ATARI computers can do either better, or easier than other small computers. The following series of programs is designed for anyone who is at least familiar with BASIC programming. What each tutorial offers is similar to an extensive magazine article with all discussion in as simple language as possible, plus you get MANY examples already typed in and running. The instruction manuals range from 10 to 50 pages, and some tutorials fill up a complete tape or disk. There is little overlap in what is taught, so anyone wanting to know all they can should buy them all (my banker thanks you). ATARI buys these from us to use in training their own people! Rave reviews have been published in ANTIC, ANALOG, CREATIVE COMPUTING, and even INFOWORLD. You trust INFOWORLD, don't you?

TT #1: DISPLAY LISTS—This program teaches you how to alter the program in the ATARI that controls the format of the screen. Normally, when you say "Graphics 8", the machine responds with a large Graphics 8 area at the top of the screen and a small text area at the bottom. Now, you will be able to mix various Graphics modes on the screen at the same time. The program does all of the difficult things (like counting scan lines). You will quickly be able to use the subroutines included in your own programs.
16K Tape or 24K Disk. **$19.95**

TT #2: HORIZONTAL/VERTICAL SCROLLING—The information you put on the screen, either GRAPHICS or TEXT, can be moved up, down, sideways, or diagonally. We provide the basic methods and leave the rest up to your skill and imagination. Includes 18 examples to get you started, with several using a small machine language subroutine for smoothness.
16K Tape or 24K Disk. **$19.95**

TT #3: PAGE FLIPPING—Now you don't have to redraw the screen every time you change the picture or text. You will learn how to have the computer draw the next screen you want to see while you are still looking at the previous screen, then flip to it instantly. You won't see it being drawn, so a complicated picture can seem to just appear. Depending on your memory size and which graphics or text modes you are using, you can instantly look at up to 50 pages. The basic method takes only 9 lines and the usefulness is infinite.
16K Tape or 24K Disk. **$19.95**

TT #4: BASICS OF ANIMATION—This program shows you how to animate simple shapes (with some sound) using the PRINT and PLOT commands, and it also has a nice little PLAYER/MISSILE GRAPHICS game you can learn from. The P/M example is explained and will get you started on this complicated subject (more fully explained in TT #5). This would be an excellent way to start making your programs come alive on the screen with movement! Recommended for beginning users.
16K Tape or 24K Disk. **$19.95**

TT #5: PLAYER/MISSILE GRAPHICS—Learn to write your own games and other animated applications! The tutorial begins with many small examples that compliment the 50 page manual, then gradually builds up to a complete game where everything you need to know is fully explained. Also included are two machine language utilities that you can use to animate Players with from BASIC. Next we include two of the best editors currently available; one for editing playfield shapes (backgrounds); and one to edit your players, and all in glorious Technicolor!! Everything except the two editors run in 16K Tape or 32K Disk. **$29.95**

TT #6: SOUND AND MUSIC—Unless you have spent many years experimenting with the four voice channels, you will learn a lot from this one! Learn to play standard notes, chords, and whole songs using some simple "tricks". One of the nicest parts are the examples of special sound effects that you can refer to whenever you need a sound for a program or to impress a friend. This program will be of interest to all ages and levels of experience!
16K Tape or 24K Disk. **$19.95**

SPECIAL DISCOUNT

Order the first six tutorials in a 3-ring binder for $99.95, a $30.00 savings!

TT #7: DOS UTILITIES—We at Educational Software have been shocked by some of the prices others are charging to offer you small utilities to help in the use of your Disk Drive. We now offer you all of the following plus explanation as to how each was written, and how to use them: A UNIQUE MENU PROGRAM, AN AUTORUN.SYS BUILDER, DISK INSPECTOR (LOOK AT SECTORS), DISK JACKET PRINTER, AUTOMATIC FORMATTER, RECORD SAVE AND LOAD UTILITY.
32K Disk Only. **$29.95**

MASTER MEMORY MAP (tm)

This book is the most valuable source of information for your ATARI you can buy. It starts out by explaining how to PEEK and POKE values into memory, so that even new computer owners can use many of these "Tricks". Then you are given 32 pages of the memory locations that are the most useful, along with hints on how to use many of the locations. Finally, it includes hints on problems you may be having with the computer and discusses the new Graphics modes 9 to 11. Even ATARI buys this book from us! **$6.95**

CONTACT YOUR LOCAL DEALER or ORDER BELOW

We have other fine programs for ATARI computers.
Write for a catalog.
Send us your programs to sell too!

USER SUBMITTED PROGRAMS

MINI-DATABASE/DIALER—stores and edits up to 8 lines of information such as names & addresses, phone numbers, messages, inventories, or anything you want. It has the usual sort, search, and print options, but it also has an unusual feature: If your file includes phone numbers and your phone company allows touch-tone phone signals, the program will DIAL THE PHONE NUMBER FOR YOU! 16K Tape or 24K Disk. **$24.95**

THE GRAPHICS MACHINE—allows the ATARI to act like more expensive graphics computers using simple commands like line, box, circle, polygon, fill, and savescreen to get a high resolution picture you can save on disk in only five seconds! Many more features! 48K Disk Only. **$19.95**

BOB'S BUSINESS—14 small business type programs accessed from a common menu. 16K Tape or 32K disk. **$14.95**

MINI WORDPROCESSOR—A simple text editor to write, save, and print several pages at a time. 32K Tape or Disk. **$19.95**

KID'S #1—Includes a MATH QUIZ, a children's TREASURE HUNT, and a DIALOGUE program. 16K Tape or 24K Disk. 3 for . . . **$14.95**

KID'S #2—SPELLING BEE, WORD SCRAMBLE, and TOUCH. 16K Tape or 24K Disk. 3 Educational games for . . . **$14.95**

PLAYER PIANO—Turns your keyboard into a mini-piano and more. 24K Tape or 32K Disk. **$14.95**

DOG DAZE—Two cute little doggies race for the fire hydrants, shoot their bones, and just have a lot of fun! A fast action program for all ages. 8K Tape or 16K Disk, in machine language. **$16.95**

GRAPHIC SYMBOL LABELS—for your keyboard to remind you of the built-in Graphics symbols. 2 complete sets for . . . **$2.95**

OUR NEWEST PROGRAMS

*** OUR BEST GAME ***

SPACE GAMES—Our family is being attacked by ALIENS, and only you can save us. A comic book manual will guide you through three games that test your ability in space skills. Includes ALIENS, SURVIVE, and ROBOT ATTACK, and is for all ages. The first two games require 16K for Tape. The last game and all Disk users need 32K. **$24.95**

MATHS FOR FUN—Another ENGLISH import teaching basic math skills. Very colorful and enjoyable to use. For ages 5 to 16. 16K Tape or 24K Disk. **$19.95**

TT #10: SOUND EFFECTS—From laser blasts to ringing phones, this tutorial will show you how to make unique sound effects in all of your programs! 16K Tape or 24K Disk. **$19.95**

MARATHON—This is a unique math quiz for one or two players. You are in a race to move your runner across the screen first! There are four levels of play with five modes of operation for each. The game uses joysticks for all input, so play is easy for young children. This wonderful learning tool is imported from ENGLAND for your learning pleasure. Your kids will never even notice they are playing an EDUCATIONAL program. 16K Tape or 24K Disk. **$19.95**

TT #8: CHARACTER GRAPHICS—Character Graphics is the best way to animate your ATARI! Make letters look like space monsters, gunfighters, or a myriad of other shapes. Use our editor to create these multicolor shapes and then we'll show you how to move them around the screen. This tutorial even shows how our Space Games were written! 16K Tape or 24K Disk. **$19.95**

To order COD, VISA or MasterCard call — (408) 476-4901
By mail include $3.00 postage (USA only) or $1.50 Memory Map only—
California residents add 6.5% TAX.—Specify Tape or disk.
For Orders Call 800-692-9520

Educational Software inc.
4565 Cherryvale Avenue Dept. CP
Soquel, Ca. 95073
(408) 476-4901

```
3 ? :? "ENTER FILE NAME BELOW - SCREE
  N WILL CLEAR then":? "ENTER L TO GE
  T PICTURE FROM DISK FILE or"
4 ? "ENTER R TO RUN PICTURE PROGRAM t
  hen"
5 ? "ENTER E TO STOP PICTURE PROGRAM
  (OPTIONAL) then"
6 ? :? :? "WHEN PICTURE IS DONE ;"
7 ? "ENTER S TO SAVE IT ON A FILE and
  /or"
8 ? "ENTER R TO RESTART PROGRAM"
9 ? :? :? "BUT FIRST ENTER NAME OF FI
  LE WHERE PICTURE IS TO BE FOUND OR
  STORED as D:NAM  or C: =":INPUT FN$
10 REM .. GRAPHICS MODE FOR PICTURE D
   ISPLAY MUST BE ISSUED HERE BEFORE
   POSSIBLE DISK FILE READ
11 GRAPHICS 24:COLOR 1:RESTORE
12 FOR J=1536 TO 1558:READ A:POKE J,A
   :NEXT J
14 IF (PEEK(764)=40) THEN IN408=8:POK
   E 764,255:GOTO 20
16 IF (PEEK(764)=0) THEN IN408=4:GOTO
    320
18 GOTO 14
20 REM ...PROGRAM THAT PUTS A NEW PIC
   TURE ON THE TV SCREEN GOES HERE EX
   CEPT SEE LINE 10
23 REM MICRO TECH UNLIM. AD IN 11/81,
   ,,,,,   COMPUTE
25 P=160:Q=100
30 XP=144:XR=1.5*3.1415927
40 YP=56:YR=1:ZP=64
50 XF=XR/XP:YF=YP/YR:ZF=XR/ZP
60 FOR ZI=-Q TO Q-1 STEP 1
70 IF ZI<-ZP OR ZI>ZP THEN GOTO 150
80 ZT=ZI*XP/ZP:ZZ=ZI
90 XL=(SQR(XP*XP-ZT*ZT))
93 XL=INT(0.5+XL)
100 FOR XI=-XL TO XL STEP 1
105 TRAP 120
110 XT=SQR(XI*XI+ZT*ZT)*XF:XX=XI
120 YY=(SIN(XT)+0.4*SIN(3*XT))*YF
130 GOSUB 170
140 NEXT XI
145 IF PEEK(764)=42 THEN IN408=8:GOTO
     300
150 NEXT ZI
160 GOTO 300
170 X1=(XX+ZZ+P)
180 Y1=YY-ZZ+Q:Y1=191-Y1
182 IF X1<0 OR X1>319 THEN RETURN
184 IF Y1<0 OR Y1>191 THEN RETURN
195 COLOR 1:PLOT X1,Y1
200 IF Y1>=190 THEN RETURN
210 COLOR 2:PLOT X1,Y1+1:DRAWTO X1,19
    1
220 RETURN
230 REM ... PROGRAM TO PUT PICTURE ON
     TV SCREEN ENDS HERE
300 IF PEEK(764)=40 THEN 2
310 IF PEEK(764)<>62 THEN 300
320 POKE 764,255:OPEN #3,IN408,0,FN$
325 POKE 891,128:REM SET SHORT Inter-
    Record Gaps FOR CASSETTE I/O
330 TVAT=PEEK(560)+PEEK(561)*256
340 RAMTOP=PEEK(106)*256
350 TVSIZ=RAMTOP-TVAT
370 SIZHI=INT(TVSIZ/256)
380 SIZLO=INT(TVSIZ-256*SIZHI)
390 TVAHI=INT(TVAT/256)
400 TVALO=INT(TVAT-256*TVAHI)
430 POKE 884,TVALO
440 POKE 885,TVAHI
450 POKE 888,SIZLO
460 POKE 889,SIZHI
500 RES=USR(1536,IN408+3)
510 CLOSE #3:POKE 764,255
520 IF PEEK(764)=40 THEN 2
525 GOTO 520
530 DATA 104,201,1,208,10,104,104,141
    ,114,3,162,48,32,86,228,133,213,1
    69,0,133,212,96,0
```

©

For Upgrade and 4.0 BASIC PET/CBM, String Thing solves the problems created by INPUT# and is faster.

Easy File Input: The String Thing

Jim Butterfield
Associate Editor

Files are easy to handle, but sometimes the INPUT# statement is too clever. It looks so carefully at the material coming in from the file that it can overprocess your information. The INPUT# statement:

- — trims spaces from the front of the data;
- — trims quotation marks from front and back;
- — gives trouble if you want to input over 80 characters;
- — drops commas and following information; and
- — drops colons and following information.

You often don't want such things to happen when you are reading a file. But the alternative is to use GET# statements, and they are slow.

Some years ago, Bill McLean, of B.M.B. Compuscience, wrote a String Thing to get around these problems and speed up input. It worked well on Upgrade ROMs, but the transition to 4.0 systems was uncomfortable; strings are stored in a different way in the newer BASIC, and the code needed major surgery. Extra code is needed in order to avoid falling prey to a berserk garbage collection routine. This made the job rather complex and called for two different versions for the two different ROM sets.

New And Improved?

It seems to be time to unearth a new String Thing, one that will work without change on both Upgrade and 4.0 BASIC PETs and CBMs. This way, your program can still move between machines without difficulty. But there's a problem: since different BASIC versions store strings in different ways, how can we make one program compatible with all?

The trick is this: instead of trying to build a new string, we'll re-use an old one. We must be careful: if the string we are recycling is only ten characters long, we must be sure we don't try to put 11 new characters into it.

How To Use It

The program listing comes in two parts: setting up the String Thing and using it. There are two things we need to do in order to set up: define the program's first variable as a string (in the example, A$), making sure that it's long enough to hold any input that we might want to catch (in this case, up to 255 bytes); and then POKEing the String Thing program into place. This setup takes place in lines 70 to 260.

Now that we have String Thing in place, we need to use it. That's the easy part: we just give SYS 896, and the program performs the equivalent of INPUT#1,A$ without the problems of INPUT. You may remember that we set A$ to a very long string; it will keep its length, but we can find out how many characters have been read by checking PEEK(139). String Thing uses location 139 to record how many characters it has received. If the first thing to come from the file is a RETURN character, this value will be zero, indicating no data characters received. On the other hand, if we fill the string space completely and still have not seen a RETURN, we'll stop at that point. The next call to String Thing will get more of the same sequence.

Some usage hints: Try to leave a string that is at least one character longer than the data you expect. If PEEK(139) ends up equal to the string length, we haven't seen the RETURN character yet – better to leave extra room. Remember that all the things that happen with an INPUT# will happen with String Thing, such as ST signalling end of file. Don't try to change your string variable (in this case, A$) as the program runs; copy the information out to another variable if you need it, e.g., X$ = LEFT$(A$,PEEK(139)). String Thing isn't location sensitive; you can move it to some other location with little trouble.

String Thing will work correctly in reading files from cassette tape or disk. Just change the OPEN statement to suit. You won't notice the speed advantage on tape, of course, but you may still benefit from the improved logic handling. String Thing works splendidly with Relative disk files. Position to the record you want in the usual way, with RECORD#1, and then substitute the SYS statement for the INPUT#1.

Try the following demonstration program. You can change the file name in line 400 to any sequential file of your own, or you can write a demonstration file using the following direct statements:

COMPUTE! Subscriber Services

Please help us serve you better. If you need to contact us for any of the reasons listed below, write to us at:

COMPUTE! Magazine
P.O. Box 5406
Greensboro, NC 27403

or call the Toll Free number listed below.

Change Of Address. Please allow us 6-8 weeks to effect the change; send your current mailing label along with your new address.

Renewal. Should you wish to renew your **COMPUTE!** subscription before we remind you to, send your current mailing label with payment or charge number or call the Toll Free number listed below.

New Subscription. A one year (12 month) US subscription to **COMPUTE!** is $20.00 (2 years, $36.00; 3 years, $54.00. For subscription rates outside the US, see staff page). Send us your name and address or call the Toll Free number listed below

Delivery Problems. If you receive duplicate issues of **COMPUTE!**, if you experience late delivery or if you have problems with your subscription, please call the Toll Free number listed below.

COMPUTE!
800-334-0868
In NC 919-975-9809

```
OPEN 1,8,3,"file,s,w"
PRINT#1,"line 1, with comma"
PRINT#1,"mission: impossible"
PRINT#1,CHR$(34);"quotes";CHR$(34)
FOR j = 1 to 40:PRINT#1,j;:NEXT j:PRINT#1
PRINT#1,"      spaced out      "
CLOSE 1
```

If you like, try getting this DATA back using INPUT# statements.

Now for String Thing:

```
70 REM ** STRING THING (PET/CBM)     **
75 REM ** UPGRADE AND 4.0 BASIC      **
80 REM **       JIM BUTTERFIELD      **
90 REMARK:  STRING MUST BE FIRST VARIABLE
100 A$="ABCDEFGHIJKLMNOPQ"
110 A$=A$+A$+A$+A$+A$
120 A$=A$+A$+A$
130 REM ABOVE SETS STRING FOR MAX (255)
200 DATA 160,2,177,42,153,134,0,200,192,6
210 DATA 208,246,162,1,32,198,255
220 DATA 32,228,255,201,13,240,11,164,139,145
230 DATA 137,200,132,139,196,136,208,238,76,20
    4,255
250 FOR J=896 TO 933:READ X:POKE J,X:T=T+X:NEX
    T J
260 IF T <>5517 THEN STOP
400 OPEN 1,8,2,"FILE"      (OR DOPEN#1,"FILE")
410 REM:  NEXT SYS SAME AS 'INPUT#1,A$'
420 SYS 896
425 REM:   l=SIZE OF INPUT (COULD BE 0)
430 L=PEEK(139)
440 PRINT LEFT$(A$,1)
450 IF ST=0 GOTO 420
460 CLOSE 1                  (OR DCLOSE)
```

Alternate Versions

For VIC and Commodore 64 machines, we may once again choose between cassette tape and disk. I've written the disk version below; to get a cassette version, you'll need to do a little juggling. That's because the String Thing program sits in the cassette buffer; it will need to be moved elsewhere if you need to use tape.

Now for the String Thing program:

```
70 REM **        STRING THING          **
75 REM **   VIC AND COMMODORE 64       **
80 REM **     JIM BUTTERFIELD          **
90 REM STRING MUST BE FIRST VARIABLE
100 A$="ABCDEFGHIJKLMNOPQ"
110 A$=A$+A$+A$+A$+A$
120 A$=A$+A$+A$
130 REM ABOVE SETS STRING FOR MAX (255)
200 DATA 160,2,177,45,153,137,0,200,192,6
210 DATA 208,246,162,1,32,198,255
220 DATA 32,228,255,201,13,240,11,164,142,145
230 DATA 140,200,132,142,196,139,208,238,76,20
    4,255
250 FOR J=896 TO 933:READ X:POKE J,X:T=T+X:NEX
    T J
260 IF T<>5535 THEN STOP
400 OPEN 1,8,3,"FILE"
410 REM:  NEXT SYS SAME AS 'INPUT#1,A$'
420 SYS 896
425 REM:   l=SIZE OF INPUT (COULD BE 0)
430 L =PEEK(142)
440 PRINT LEFT$(A$,1)
450 IF ST=0 GOTO 420
460 CLOSE 1
```

If you have an Original ROM machine, you can't handle disk. Even tape files have certain problems. If you plan to write and read files, you will be much better off if you upgrade your machine to Upgrade ROM. This can be done with a chip change. Even so, let's show that String Thing can be made to work here: we'll read a tape on an Original ROM system.

You may write a demonstration file TO tape in exactly the same way as before, except that you must change the OPEN statement TO:

```
OPEN 1,1,1,"file"
```

```
70 REM **        STRING THING          **
75 REM **     ORIGINAL ROM BASIC       **
80 REM **      JIM BUTTERFIELD         **
90 REM STRING MUST BE FIRST VARIABLE
100 A$="ABCDEFGHIJKLMNOPQ"
110 A$=A$+A$+A$+A$+A$
120 A$=A$+A$+A$
130 REM ABOVE SETS STRING FOR MAXIMUM (255)
200 DATA 160,2,177,124,153,216,0,200,192,6
210 DATA 208,246,162,1,32,198,255
220 DATA 32,228,255,201,13,240,11,164,221,145
230 DATA 219,200,132,221,196,218,208,238,76,20
    4,255
250 FOR J=896 TO 933:READ X:POKE J,X:T=T+X:NEX
    T J
260 IF T<>6009 THEN STOP
400 OPEN 1,1,0,"FILE"
410 REM:  NEXT SYS SAME AS 'INPUT#1,A$'
420 SYS 896
422 REM:  ST LOGIC CHANGE
423 IF ST=0 GOTO 460
425 REM:  l=SIZE OF INPUT (COULD BE 0)
430 L=PEEK(221)
440 PRINT LEFT$(A$,1)
450 GOTO 420
460 CLOSE 1
```

Assembly Listing

For those who would like to track the machine language, here's the assembly version of the program. The version is Upgrade/4.0 ROMs (the first BASIC program).

```
0380  A0 02          LDY #2            Copy string..
0382  B1 2A    LOOP1 LDA (VARTAB),Y    info TO work
0384  99 86 00       STA WORK,Y        area
0387  C8             INY               ..Four bytes
0388  C0 06          CPY #6
038A  D0 F6          BNE LOOP1
038C  A2 01          LDX #1            Connect file #1
038E  20 C6 FF       JSR CHKIN         ..as input.
0391  20 E4 FF LOOP2 JSR GETIN         Get character
0394  C9 0D          CMP #$0D          Return?
0396  F0 0B          BEQ QUIT          Yes, quit
0398  A4 8B          LDY COUNT         No, get pointer
039A  91 89          STA (STRING),Y    ..and stash char
039C  C8             INY               Count it
039D  84 8B          STA COUNT         Save count
039F  C4 88          CPY LENGTH        Full string?
03A1  D0 EE          BNE LOOP2         Nope, do more
03A3  4C CC FF       JMP CLRCHN        Disconnect & quit
```

String Thing solves many file input problems: in particular, long data blocks and special characters become very simple. It's as fast as INPUT, but for most purposes it's better.

Programming The PET/CBM

by Raeto Collin West

The book described by Jim Butterfield as

"...unquestionably the most comprehensive and accurate reference I have seen to date..."

The Reference Encyclopedia for Commodore 2000, 3000, 4000, and 8000 series computers and peripherals.

Here's just a sample of reviewer and reader reaction:

From reviewers:

Educational Computing Review by Stephen Potts
"Of all the books I have read on the PET this book *Programming the PET/CBM* by Raeto West must rank as one of the most comprehensive and readable accounts on the PET that I have ever had the pleasure to see...

"If you wish to get more from your PET than arcade games and simple teaching programs then this book is a must for your bookshelf. It does not matter whether you run on BASIC 1, BASIC 2, or BASIC 4 since all routines are supplied with addresses and changes to make them run on any machines wherever possible...

"...this book, with its lucid explanations of the PET, its useful routines and programming hints, is an essential purchase."

IPUG Magazine Review (British PET User Group) by Ron Geere
"This publication represents over a year's intensive research ... and the resulting product is a valuable work of reference. A tremendous amount of useful information has been packed in this 500+ page work at which I was so over-awed that I did not know how to start this review at first...

"This book is a must for every CBM/PET user."

From readers:

"...a book the average to advanced user cannot afford *not* to possess..."

"My copy of your *'Programming the PET/CBM* has been in daily use for nearly a month and I am finding it totally addictive, suffering severe withdrawal symptoms whenever I try half-heartedly to move on to other reading matter. It is without doubt the best book on its subject available today..."

"I have recently acquired a copy of your book *Programming the PET/CBM* and must congratulate you on its concept and on packing in so much detail. It's so very much better than anything I have had up to now that it'll be my constant reference manual."

"I have received my copy of *Programming the PET/CBM* by Raeto West and I have recommended it to several of my students. This book is so valuable that I cannot now afford to be without it."

Published exclusively in North America by **COMPUTE! Books**. The book is an astonishing reference manual of useful information. Contents include this and much more:

1 **Introduction and overview:** Plan of the book, sources of information, features and chronology of CBM hardware.
2 **BASIC and how it works:** Storage of BASIC and its variables; tokens, pointers, syntax; optimising BASIC.
3 **Program and system design:** Capabilities of the equipment; charts, algorithms, space, timing.
4 **Effective programming in BASIC:** Seventeen examples, including subroutines, dates, DATA, INPUT, rounding.
5 **Alphabetic reference to BASIC keywords:** Full descriptions, with examples, of all keywords, with methods for adding additional commands not present in CBM BASIC, e.g. AUTO, DEL, OLD, POP, PRINT USING, SORT, VARPTR.
6 **Disk drives:** Descriptions of operation and workings of disk drives, with BASIC and machine-code examples; bugs.
7 **Alphabetic reference to disk BASIC commands:** BASIC 4 disk commands with examples and notes.
8 **Other peripherals and hardware:** Tape storage and handling; printers; modem; keyboard; EPROMs; reset switches.
9 **Graphics and sound:** Tables of CBM characters; CRT chip; animation, bar plots, 80 by 50 etc.; user-port sound.
10 **The transition to machine-code:** Introductory concepts; a BASIC monitor; use of MLM, Supermon, Extramon; easy examples.
11 **More 6502 machine-code:** 6502 hardware features; eighteen common problems in programming; debugging.
12 **Alphabetic reference to 6502 opcodes:** Examples, notes, and explanations on each opcode from ADC to TYA.
13 **Using ROM routines:** IRQ, NMI, RESET; the Kernel; examples – modifying LIST; ordinary and relocating loaders.
14 **Effective 6502 programming:** Assemblers; CHRGET and wedges; PIAs, VIA, IEEE; common mistakes.
15 **Index to BASIC ROMs and RAM:** Memory map of RAM and ROM, detailing and comparing BASICs 1, 2, and 4.
16 **Mathematical programming:** Precision; equations; statistics; simulation; finance; calculus; machine-code.
17 **Programming in business and education:** Examples, applications and pitfalls in business and education.
Appendices: 6502 reference charts; Supermon listings; ASCII; glossary

Plus many programs, diagrams and charts. Paperback, 504 pages. ISBN 0 942386 04 3. **$24.95.**

To Order

Programming The PET/CBM

Call

TOLL FREE 800-334-0868
In NC Call 919-275-9809

Or send coupon to

COMPUTE! Books, P.O. Box 5406, Greensboro, NC 27403

In England, order from Level Limited, P.O. Box 438, Hampstead, London, NW3 1BH. Price in England is £14.90, including P & P.

Please send ________ copy (copies) of **Programming The PET/CBM** at $24.95 each. (In the US and Canada, add $3.00 shipping and handling. Outside North America add $9.00 for air mail delivery, $3.00 for surface delivery).

All orders must be prepaid in US funds (money order, check, or charge).

☐ Payment Enclosed

Please charge my ☐ VISA ☐ MasterCard ☐ Am. Express

Account No. ________ Expires ___ /

Name ________

Address ________

City ________ State ________ Zip ________

Country ________

Allow 4-6 weeks for delivery. Foreign surface delivery allow 2-4 months.

VIC machine language programmers: here's one of the most valuable tools there is for working in machine language. Thirty-four commands are at your disposal including single-step, hex conversion, search, EPROM routines, and a relocator. If you aren't yet working with machine language, the instructions for entering and using this program are easy to follow and you might want to add it to your library for later use. Micromon might even lead you into the intriguing and often quite useful world of the machine's own language. As presented, this program takes up 4K of memory from $4000 (16384 decimal) to $4FFF (20479), but there are instructions for locating it elsewhere in RAM memory.

VIC Micromon

Bill Yee
Winnipeg, Manitoba

I could not resist F. Arthur Cochrane's suggestion in the January 1982 **COMPUTE!** about modifying the PET Micromon for VIC use and publishing the results. The VIC Micromon presented here is the product of several months of use and revision of an initial modified version completed in April 1982.

The initial modifications involved redefining Micromon's workspace from the PET tape buffer at $27A to the VIC tape buffer at $33C. The VIC tape buffer was redefined to $375, and the Micromon Load and Save commands rewritten to read and write absolute memory images to cassette tape. A Verify command was added as well.

Other modifications included redefining locations used by Micromon in the BASIC and kernal storage areas from the PET to the VIC equivalents. The address of the decimal output routine used by the conversion commands was changed to the VIC routine at $DDCD. Because the VIC IRQ vector is at $314, rather than at a zero page location such as $90 for the PET, the single instruction interrupt time had to be increased to compensate for the absolute store instructions used. Two of these instructions are used in setting the IRQ vector when rolling out of Micromon for the Walk and Quick trace commands. Also, hardware differences required changes to the interrupt timer addresses. Originally at $E848 and $E849 for the PET, these were redefined as $9128 and $9129 for the VIC.

Micromon uses an IRQ service routine to provide forward and reverse scrolling for the Hex conversion, Memory dump, and Disassembly commands. Because VIC's screen readily relocates and has a unique format of 23 lines by 22 columns, most of the scrolling code in the IRQ service routine had to be rewritten.

Solving Early Problems

I had previously built a 16K memory board for operation in the $2000 to $5FFF address space of the VIC. So, along with the modifications already mentioned, the programming for the initial VIC Micromon included a relocation from $1000 to $4000. The code was entered as two separate 2K blocks on my VIC using a BASIC Hex Editor and subsequently programmed onto two 2716 EPROMs with a programmer I had built to operate off the VIC USER I/O port. An extension of the BASIC Hex Editor provided the EPROM programming control.

Once installed on my VIC at $4000, VIC Micromon was accessed by a SYS16384 from BASIC. Each Micromon command was exercised for proper operation. I had difficulty returning to BASIC with the E and X commands, as well as stack interference in executing test programs with the G, W, and Q commands. I solved these problems by changing the stack area used by VIC Micromon to the bottom half of the $100 page. This was done by setting the stack pointer to $7F rather than $FF in the command input section of VIC Micromon.

The New locater command would work only once in the word mode when the command was invoked several times consecutively. A flag at $28C for PET Micromon and $34E for VIC Micromon differentiates between the instruction mode and the word mode in the New locater command. The problem is due to the flag being used also as an offset index to check absolute addresses. The flag was being incremented without being cleared on each consecutive New locater command in word mode. This was fixed by inserting code in the command input section of VIC Micromon to always clear $34E prior to execution of any command.

The Assembler command would not properly assemble branch instructions which had branch offsets of $7E and $7F. This problem resulted from a simple plus/minus range check prior to adjusting the offset value by -2. The fix for VIC Micromon is given here as an inline patch for PET Micromon users. The PET Micromon code from

VICWARE

by MicroSpec Ltd.

Colorful, Exciting Software for Commodore Computers!

Moon Lander	9.95	*Disk Data Base Mgr.	59.95
Reversal 1	9.95	*Cassette Data Mgr.	19.95
Bricks	9.95	*Auto Expense	19.95
Blackjack	9.95	Home Finance	9.95
Tennis	9.95	*Reversal 2	13.95
Slots	9.95	*Home Inventory	16.95
Bombs Away	9.95	*Portfolio Mgr.	19.95
*Biorhythm	9.95	Alien Invaders	9.95
Wrap	9.95	Math Drill	9.95
Starfighter	13.95	*Hangman	12.95
Capture	9.95	*Spelling Gr. 2-6	9.95
Typing Drill	9.95	***Req. Memory Expansion**	

Dust Covers - VIC 20—64 . . . . 5.95
All other CBM's 11.95

A full compliment of games and business programs is also available for the CBM 64 and other Commodore computers.

Send self-addressed, stamped envelope for FREE catalogue.

DEALER INQUIRIES WELCOME

2905 Ports O'Call Court
Plano, Texas 75075
(214) 867-1333

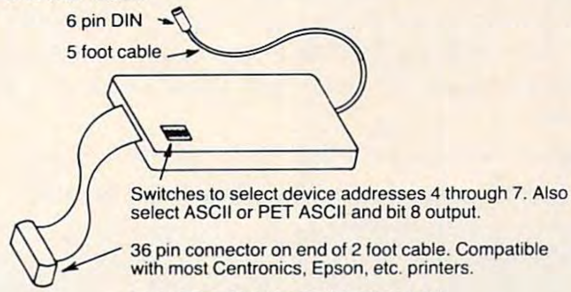
VISA and MASTERCARD Accepted

NEW CBM-64 AND VIC-20 PRODUCTS

***** CBM-64/VIC-20 PRINTER INTERFACE *****

Model MW-302 $119.95

This unit will interface your VIC-20 or CBM-64 to standard parallel printers such as Epson, Centronics, C. Itoh, and many others. Allows printing of full upper and lower case.



MW 302 PRINTER INTERFACE

***** SYSTEM 310 ***** $89.95

Consists of CBM-64, MX-80, Printer interface MW-302, and C2N cassette
(CBM-64 sold separately for $595)

***** DIGITAL TO ANALOG CONVERTER OR 8 CHANNEL ANALOG TO DIGITAL CONVERTER *****

Model MW-304v $129.95

Allows your VIC or CBM to output analog signals or digitize up to 8 analog channels.

MICRO WORLD ELECTRONIX INC
6340 W. Mississippi Ave.
Lakewood Colorado 80226 **(303) 934-1973**

VIC-20* HARDWARE

VIC-20	PERSONAL COMPUTER	$194.88	VIC-1111	16K RAM EXP	$ 99.95
VIC-1011A	RS232C INTERFACE	$ 39.95	VIC-1110	8K RAM EXP.	$ 49.95
VIC-1515	PRINTER	$334.95	VIC 1210	3K RAM EXP	$ 34.95
CARD? (CARDPRINT)	VIC TO CENTRONICS PARALLEL		VIC-1010	EXPANSION MODULE	$139.95
	INPUT (IE. TRSXX OR EPSON)		CARDBOARD/6	EXPANSION INTERFACE	$ 79.95
	PRINTER INTERFACE	$ 79.95		(CHRISTMAS SPECIAL)	
VIC-1530	DATASETTE	$ 64.99	CARDBOARD/3		$ 29.95
CARDETTE	UNIVERSAL CASSETTE INTERFACE		VIC-1311	JOYSTICK	$ 9.95
		$ 29.95	VIC-1312	GAME PADDLES	$ 19.95
VIC-1540	DISK DRIVE	$349.95	VIC-1600	TELEPHONE MODEM	$ 99.95

VIC-20* SOFTWARE

BUSINESS APPLICATIONS

TELEPHONE LISTER	FILE BY NAME OR KEY	$ 9.95	DATA FILES	UNLIMITED FILES	$ 14.95
APPT. CALENDAR	DON'T MISS THAT CALL	$ 16.95	HOME BUDGETEER	RUN YOURS EFFICIENTLY	$ 15.95
STOCK PORTFOLIO	COMPLETE INVEST. MGMT	$ 16.95	HOME INVENTORY	INSURANCE NECESSITY	$ 12.95
GENERAL LEDGER	BAL SHEET & INC STMT	$ 29.95	CHECKBOOK	A MUST !!!!!!!!!!	$ 12.95
INVENTORY	DOES IT ALL!!!	$ 19.95	INCOME TAX	I.R.S. BEWARE	$ 14.95
DEPRECIATION CALC	ST OR DOUB DEC. BAL	$ 10.95	CLUB LISTER	WHO'S IN? PH/ADD	$ 13.95
NET WORTH STMNT	HOW MUCH ARE YOU?	$ 14.95			

GAMES FOR ALL

SNACKMAN	A MUST !!!!	$ 14.95	EXECUTIONER	HANGMAN W/TWIST	$ 14.95
THE CUBE	IN COLOR — GREAT!	$ 14.95	BREAKOUT	THE CLASSIC IN COLOR	$ 7.95
INVASION	SAVE US (AGAIN?)	$ 14.95	GRAPHICS	SHOW OFF YOUR VIC	$ 12.95
RESCUE FROM NUFON	SUPER ADVENTURE GAME	$ 14.95	SPELIT	UP TO 20 PLAYERS	$ 14.95
TANK	TWO PLAYERS (GOTCHA)	$ 14.95	JOURNEY	TO THE BOTTOM	$ 14.95
MOTOCROSS	GO GO GO	$ 14.95	ESCAPE	3-D GRAPHICS MAZE	$ 14.95

COMPU SENSE

VIC-20® is a registered trademark of Commodore

TO ORDER:
812 S. LIGHTNER
WICHITA, KS 67218
(316) 684-4660
PERSONAL CHECKS ACCEPTED (ALLOW 3 WEEKS), OR C.O.D.
HANDLING CHARGES $1.50

$1525 to $1538 was rewritten to use a value of $82 for a range check. The resulting code, which is one byte longer, was accommodated by replacing the branch and jump instructions at $150E to $1512 with two NOPs and a branch. The code at $1513 to $1524 was moved down one location to be at $1512 to $1523, and the branch instructions originally at $1519 and $1520 had their offsets adjusted up by one.

Table 1: Micromon Locations Redefined for VIC

Location	Old Value	New Value	Location	Old Value	New Value
150E	F0	EA	1523	13	11
150F	03	EA	1524	11	90
1510	4C	D0	1525	90	01
1511	91	7F	1526	0A	88
1512	15	20	1527	98	C8
1513	20	3C	1528	D0	D0
1514	3C	18	1529	6F	6F
1515	18	AC	152A	AE	98
1516	AC	8B	152B	91	2A
1517	8B	02	152C	02	AE
1518	02	F0	152D	30	91
1519	F0	2F	152E	6A	02
151A	2E	AD	152F	10	E0
151B	AD	93	1530	08	82
151C	93	02	1531	C8	A8
151D	02	C9	1532	D0	D0
151E	C9	9D	1533	65	03
151F	9D	D0	1534	AE	B0
1520	D0	20	1535	91	03
1521	1F	20	1536	02	38
1522	20	13	1537	10	B0

F. Arthur Cochrane presented an extra set of commands implemented on about an additional 1K module called Micromon Plus. Of the commands in Micromon Plus, I felt that the Print switcher and PROM programmer commands would be desirable in VIC Micromon. However, instead of creating an extra module above the existing 4K of the initial VIC Micromon, I decided that, with some code crunching, the Print switcher and PROM programmer commands could be contained within an enhanced 4K VIC Micromon. Since there are many printers available with a RS-232 interface, the enhanced VIC Micromon Print switcher command would support the RS-232 interface on the VIC User I/O port rather than the VIC serial port interface. Also, since I already had an EPROM programmer working off the User I/O port, the enhanced VIC Micromon PROM programmer command would support my programmer rather than one on the IEEE bus.

Memory Saving

Code crunching consisted of code optimization and code removal while keeping function and structure. The table of internal Micromon addresses used to set up and check interrupt vectors was removed. While the table helped in any relocation of Micromon, I felt that direct changes of vector address values were relatively easy and well worth the program area saved. VIC Micromon has six locations with interrupt vector values that must be changed when VIC Micromon is relocated. The checks for the different versions of PET BASIC were removed as well. It didn't make sense to have universality in some parts of VIC Micromon when a good part was applicable only to the VIC.

The VIC has two levels of indirection for the input and output to device routines. The first level is the use of PET compatible addresses for the jump vectors at the end of the 8K kernal ROM. The second level of indirection is the use of an indirect jump table in RAM which allows the user to redefine the vectors. Because of this, I felt that the use of I/O jump vectors at the beginning of Micromon was unnecessary.

Besides the four ROM routines mentioned by F. Arthur Cochrane (input character, output a character, load a program, and save a program), there were other ROM routines, such as get from keyboard ($FFE4) and close I/P and O/P channels ($FFCC), used by PET Micromon. For VIC Micromon these were expanded to about 20 ROM routines, since I chose to use VIC kernal routines when possible. For example, the test for STOP key code at location $18AE of the PET Micromon was replaced with a call to the test STOP key routine at $FFE1 in the VIC kernal ROM.

Altogether, the code crunching resulted in over 250 bytes freed. This allowed me to add the Print switcher and EPROM commands along with a few handy ones which did not require much code to implement. Following is a detailed description of VIC Micromon commands with a different format or function from similar commands in PET Micromon and of new commands in VIC Micromon.

VIC Micromon Instructions

Initialize Memory And Screen Pointers

.I 1000 1E00 1E

Define low memory as $1000 and high memory as $1E00 regardless of the memory present. The screen is defined to start at the $1E page of memory. The screen memory should always be on an even

page within the range of $1000 to $1E00. Odd page values result in incorrect setup and operation of the VIC display. Although 3K of RAM can be added at $400 to $FFF, this memory is not accessible for use as screen memory.

Memory pages at $000 and $200 are accessible, but are not usable since they are used for BASIC and kernal storage, working buffers, and stack area. If the screen page is within the low to high memory range specified, there can be usage conflict of the screen memory pages. If the "I" command is used and exit is made to BASIC, the NEW command must be invoked in the BASIC environment to clean up the memory pointers used by BASIC.

Jump To Micromon Subroutine

.J 2000

The subroutine at $2000 is called while remaining in the VIC Micromon environment. The assembly language subroutine should exit by using a RTS instruction, which causes a return to the command input section of VIC Micromon. The machine image as shown by the Register display command is not used, nor is it disturbed when the subroutine returns to VIC Micromon.

Load

.L 2000 "TEST FILE" 01

Search for and, if found, load into memory the data file on device #1 named TEST FILE. If the name is not specified, the first file found is loaded. The data is loaded into memory starting at location $2000. The last address loaded is determined by the length of the binary data file. If the device number is not specified, it defaults to device #1, which is the VIC cassette tape. The original memory addresses and name of the last file read can be inspected by doing a Memory display of the tape buffer which is at $375 for VIC Micromon.

Print Switcher

.P

If the output is to the screen, then switch the output to the RS-232 channel (device #2). If the output is not to the screen, restore the output to the screen with the RS-232 channel left active until the RS-232 output buffer is drained. Note that opening the RS-232 channel grabs 512 bytes for I/O buffering from the top of memory.

.P 0000

Regardless of the output, clear the RS-232 channel and set output to the screen.

.P CCBB

If the output is to the screen, set CC into the RS-232 command register at location $294 and BB into the RS-232 control register at location $293. Output is then switched to the RS-232 channel. This command is invalid if output is not currently to the screen.

Command Register Format

Field	Use	Value	Description
7,6,5	Parity Options	- - 0	Parity disabled
		0 0 1	Odd parity
		0 1 1	Even parity
		1 0 1	Mark transmitted
		1 1 1	Space transmitted
4	Duplex	0	Full duplex
		1	Half duplex
3,2,1	Unused		
0	Handshake	0	3 line
		1	x line

Control Register Format

Field	Use	Value	Description
7	Stop Bits	0	1 stop bit
		1	2 stop bits
6,5	Word Length	0 0	8 bits
		0 1	7 bits
		1 0	6 bits
		1 1	5 bits
4	Unused		
3,2,1,0	Baud Rate	0 0 0 0	User rate
		0 0 0 1	50 Baud
		0 0 1 0	75
		0 0 1 1	110
		0 1 0 0	134.5
		0 1 0 1	150
		0 1 1 0	300
		0 1 1 1	600
		1 0 0 0	1200
		1 0 0 1	1800
		1 0 1 0	2400

Save

.S 2000 3000 "TEST FILE" 01

Save memory from $2000 up to, but not including, $3000 onto device #1, which is the VIC cassette tape. If the device number is not specified, it defaults to device #1. The name *TEST FILE* is placed in the file header for the file saved.

Verify

.V 2000 "TEST FILE" 01

Search for and verify, if found, the data file on device #1 named "TEST FILE." If the name is not specified, the first file found is verified. The data is verified by reading the file and comparing it to the data in memory starting at location $2000. If not specified, the device defaults to device #1. If there is a mismatch, the message ERROR is outputted to the screen at the end of the file verification.

Command End Tone

.(

Enable the command end tone. A continuous tone will be generated at the end of execution of the next command. The tone can be turned off but still be enabled by just hitting the carriage return. No tone is generated if there is a syntax error while inputting the next command.

.)

Disable the command end tone.

Program EPROM

.π 2800 2FFF 00

Program the 2716 type EPROM via the EPROM programmer on the VIC User I/O port with data read from memory starting at location $2800 and ending at location $2FFF. The last input parameter specifies in hex the starting 256 byte page offset on the EPROM. If the low order byte of the starting memory address is zero and the offset is zero, then the programming starts with the first byte of the EPROM. For example, to program only the last byte of the 2K EPROM with a data byte from location $2FFF in memory, the command would be:

.π 2FFF 2FFF 07

During programming, a compare of EPROM to memory is done for each data byte just after it is written to the EPROM. Any mismatch due to failure to program the EPROM results in output to the screen of the mismatched memory location. If programming must be terminated early, just hit the STOP key. No other means should be used to abort EPROM programming. *A warm restart or power down while programming can damage the EPROM.*

Read EPROM

.£ 2999 27FF 00

Load memory starting at location $2000 and ending at location $27FF with data read from the EPROM via the EPROM programmer on the VIC User I/O port. The last input parameter specifies in hex the starting 256 byte page offset on the EPROM. If the low order byte of the starting memory address is zero and the offset is zero, then the reading starts with the first byte of the EPROM. For example, to read only the last byte of the 2K EPROM and load that byte into memory at location $10FF, the command would be:

.£ 10FF 10FF 07

During memory load, a compare of EPROM to memory is done for each data byte just after it is written to memory. Any mismatch because of failure to write the memory with data from the EPROM results in output to the screen of the mismatched memory location. The STOP key can be used to terminate the command early.

Compare EPROM

.=3000 37FF 00

Compare memory starting at location $3000 and ending at location $37FF with data read from the EPROM via the EPROM programmer on the VIC User I/O port. The last input parameter specifies in hex the starting 256 byte page offset on the EPROM. If the low order byte of the starting memory address is zero and the offset is zero, then the reading starts with the first byte of the EPROM. For example, to read only the last byte of the 2K EPROM and compare that with the data byte in memory at location $37FF, the command would be:

.=37FF 37FF 07

Any mismatch between the EPROM and corresponding memory data results in output to the screen of the mismatched memory location. The STOP key can be used to terminate the command early.

Table 2: Commands for VIC Micromon

VIC Micromon Instruction	Command
SIMPLE ASSEMBLER	A
BREAK SET	B
COMPARE MEMORY	C
DISASSEMBLER	D
EXIT VIC MICROMON	E
FILL MEMORY	F
GO RUN	G
HUNT MEMORY	H
INITIAL MEMORY & SCREEN PTRS	I
JUMP TO SUBROUTINE	J
LOAD MEMORY FROM DEVICE	L
MEMORY DISPLAY	M
NEW LOCATER	N
OFFSET OR BRANCH CALCULATE	O
PRINT SWITCHER	P
QUICK TRACE	Q
REGISTER DISPLAY	R
SAVE MEMORY TO DEVICE	S
TRANSFER MEMORY	T
VERIFY MEMORY FROM DEVICE	V
WALK CODE	W
EXIT TO BASIC	X
ASCII CONVERSION	"
DECIMAL CONVERSION	#
HEXADECIMAL CONVERSION	$
BINARY CONVERSION	%
CHECKSUM MEMORY	&
COMMAND END TONE ENABLE	(
COMMAND END TONE DISABLE	)
ADDITION	+
SUBTRACTION	-
LOAD MEMORY FROM EPROM	£
PROGRAM EPROM FROM MEMORY	π
COMPARE EPROM TO MEMORY	=

Of the set of commands available on the PET version of Micromon, only two were removed in the conversion to the VIC. These were the K (Kill Micromon) and Z (change character sets) commands. The K command is not necessary since the VIC doesn't have the TIM monitor. The function of the Z command, which is to change character sets, is already provided for on the VIC by pressing the VIC shift and Commodore keys at the same time. The rest of the commands described by F. Arthur Cochrane for the PET Micromon (see **COMPUTE!**, January 1982, p. 160) all apply identically to the commands for VIC Micromon, with the exception of the LOAD and SAVE commands, which have different formats.

VIC Micromon is always entered from VIC BASIC by a SYS16384 when it resides at $4000 to $4FFF. Either the E (Exit VIC Micromon) or the X (Exit to BASIC) command would be used to exit VIC Micromon and return to the BASIC environment. The difference between these two commands is that the X command leaves the VIC Micromon vectors in the IRQ and BRK interrupt vector locations while in the BASIC environment. Also, the tape buffer is left defined as beginning at $375. Thus, certain IRQ interrupt conditions such as the moving of the cursor to the top or bottom of the screen with output from a D, M, or $ command displayed will cause scrolling and reentry into VIC Micromon. Also, if a BRK instruction is executed, VIC Micromon will be reentered via its BRK interrupt handler.

The E command restores the IRQ and BRK interrupt vectors and resets the tape buffer pointer to a value of $33C prior to exit to the VIC BASIC environment. Thus all active linkages and vectors to VIC Micromon are removed, and the VIC behaves as if VIC Micromon never existed. In particular, the E command should be used to exit VIC Micromon when the normal VIC cassette tape LOAD, SAVE, and VERIFY commands are to be used in the BASIC environment. Otherwise, invalid results are likely to occur with some tape operations.

Both the E and X commands expect the stack pointer value (as shown for SP by the Register display command) to be the same as when VIC Micromon was first entered via the BASIC SYS command. If the value of SP or the part of the stack pointed to by SP is overwritten, such as by the execution of faulty code, a clean exit to BASIC by the E and X commands is unlikely. However, both the E and X commands do check if BASIC has been initialized, and if not, exit to BASIC is via an indirect jump to the address given at location $C000. The address given in location $C000 is $E378, which is the entry to initialize BASIC. In this case, the value of SP and the contents of the stack aren't important. Once in BASIC and regardless of how the exit from VIC Micromon was made, any subsequent access to VIC Micromon at $4000 is always by a SYS16384.

VIC Micromon as given here is located from $4000 to $4FFF. It can be relocated to any 256 byte page boundary by making the changes, as shown in the following example, which relocate VIC Micromon from $4000 to $6000.

The example begins with VIC Micromon at $4000 and ends with a relocated VIC Micromon in RAM at $6000 as well as the original at $4000.

```
.T 4000 4FFF 6000
.N 6000 6003 2000 4000 4FFF
.N 6012 6E6D 2000 4000 4FFF
.N 6FB5 6FFE 2000 4000 4FFF W
```

Location	Old Value	New Value
6018	45	65
602A	43	63
6392	4C	6C
6650	45	65
66E7	45	65
6897	43	63

In order to access the relocated VIC Micromon at $6000, exit using the E command and then from BASIC use SYS24576.

Cartridge And Checksum

The VIC-20 treats cartridge programs located at $A000 in a special way. On power-up, a test is made for the existence of the $A000 cartridge program, and if one exists, an indirect jump is made to the address specified at location $A000. This jump is made after the stack pointer is initialized, but before anything else is done. Because kernal initialization has not occurred, any cartridge program using kernal I/O routines must do kernal initialization before using those routines.

VIC Micromon as presented here has the kernal initialization calls built in so that it can easily be relocated and used as a cartridge program at $A000. Besides making the changes to relocate it to $A000, the only additional changes are to the first four bytes of VIC Micromon.

Location	Contents
A000	09
A001	A0
A002	C7
A003	FE

Power-up with VIC Micromon installed as a cartridge at $A000 will result in immediate entry into VIC Micromon. Because BASIC is not initialized when the E or X command is used after power-up, the exit to BASIC will be via an indirect jump to the address given in location $C000, which is the entry to initialization of BASIC. Once in BASIC, subsequent access of VIC Micromon at $A000 must be made to location $A012, which is done via a SYS40978.

There is one last point, or rather one last byte, in VIC Micromon which is not used for anything other than to make the 4K byte checksum of VIC Micromon come out to a rounded up page value. For example, the VIC Micromon from $4000 to $4FFF has a data byte value of $E6 at location $4FFF that results in a checksum of $BF00. This provides an easy way to verify the integrity of VIC Micromon without having to memorize or look up a checksum.

Program 1.

```
1 IFPEEK(20478)=67ANDPEEK(20479)=230THENRUN10
2 PRINT"VIC20 MICROMON LOAD &":PRINT"VERIFIC
    ATION PROGRAM.":PRINT
3 PRINT"BILL YEE / 27 AUG 82":PRINT:PRINT"AT
     LEAST 4K BYTES OF":PRINT"RAM MUST BE
     INSTALLED"
4 PRINT"AT 16384 ($4000) ELSE":PRINT"LOAD WI
    LL FAIL.":PRINT
5 PRINT"IF LOADED & VERIFIED":PRINT"OK, MICR
    OMON WILL BE":PRINT"ENTERED AUTOMATIC
    ALLY."
6 LOAD"",1,1
10 DATA 12915,16867,15061,13732,14507,13829,1
    3801,12813,15027,13920
20 DATA 14355,11977,11877,13583,11338,15173,1
    2337,14852,14051,15723
30 DATA 13442,14047,14746,15059,13134,15848,1
    5858,17856,13327,8655
40 DATA 12171,10231
100 Q=16384
110 FOR BLOCK=1TO32
120 FOR BYTE=0TO127
130 X=PEEK(Q+BYTE):CK=CK+X
140 NEXT BYTE
150 READ SUM
160 IF SUM <> CK THEN PRINT"ERROR IN BLOCK #"B
    LOCK:ERR=1:GOTO170
165 PRINT"BLOCK #"BLOCK" OK"
170 CK=0:Q=Q+128
180 NEXT BLOCK
190 IFERR=1THENPRINT"LOAD FAILED":END
200 SYS16384
```

Program 2.

```
4000 78 4C 12 40 41 30 C3 C2
4008 CD 20 8D FD 20 52 FD 20
4010 18 E5 20 19 43 A9 DF A2
4018 45 8D 16 03 8E 17 03 AD
4020 14 03 AE 15 03 C9 91 D0
4028 04 E0 43 F0 09 8D 60 03
4030 8E 61 03 20 94 48 A9 75
4038 85 B2 A9 80 8D 8A 02 85
4040 9D A2 D6 20 5D 4E 8E 48
4048 03 8E 64 03 58 00 CE 3D
4050 03 D0 03 CE 3C 03 20 AE
4058 45 A2 42 A9 2A 4C 3D 49
4060 A9 3F 20 D2 FF A9 00 2C
4068 A9 0F 8D 0E 90 20 AE 45
4070 A9 2E 20 D2 FF A9 00 8D
4078 4E 03 8D 56 03 8D 64 03

4080 A2 7F 9A 20 8C 48 C9 2E
4088 F0 F9 C9 20 F0 F5 A2 24
4090 DD 90 4F D0 13 8D 49 03
4098 8A 0A AA BD B5 4F 85 FB
40A0 BD B6 4F 85 FC 6C FB 00
40A8 CA 10 E5 4C 60 40 A2 02
40B0 D0 02 A2 00 B4 FB D0 09
40B8 B4 FC D0 03 EE 56 03 D6
40C0 FC D6 FB 60 A9 00 8D 4E
40C8 03 20 13 42 A2 09 20 38
40D0 49 CA D0 FA 60 A2 02 B5
40D8 FA 48 BD 53 03 95 FA 68
40E0 9D 53 03 CA D0 F1 60 AD
40E8 54 03 AC 55 03 4C F4 40
40F0 A5 FD A4 FE 38 E5 FB 8D
40F8 53 03 98 E5 FC A8 0D 53

4100 03 60 A9 00 F0 02 A9 01
4108 8D 57 03 20 CB 47 20 AE
4110 45 20 F0 40 20 21 48 90
4118 18 20 E7 40 90 7F 20 59
4120 41 E6 FD D0 02 E6 FE 20
4128 1F 49 AC 56 03 D0 6E F0
4130 E8 20 E7 40 18 AD 53 03
4138 65 FD 85 FD 98 65 FE 85
4140 FE 20 D5 40 20 59 41 20
4148 E7 40 B0 51 20 AE 40 20
4150 B2 40 AC 56 03 D0 46 F0
4158 EB A2 00 A1 FB AC 57 03
4160 F0 02 81 FD C1 FD F0 0B
4168 20 F8 47 20 38 49 20 E1
4170 FF F0 2A 60 20 E6 47 20
4178 A1 49 F0 1E AE 56 03 D0

4180 1C 20 F0 40 90 17 60 20
4188 54 48 8D 4B 03 20 7C 41
4190 AD 4B 03 81 FB 20 1F 49
4198 D0 F3 4C 60 40 4C 68 40
41A0 20 74 41 20 8C 48 C9 27
41A8 D0 12 20 8C 48 9D 65 03
41B0 E8 20 A4 49 F0 20 E0 20
41B8 D0 F3 F0 1A 8E 59 03 20
41C0 5F 48 90 D6 9D 65 03 E8
41C8 20 A4 49 F0 09 20 57 48
41D0 90 C8 E0 20 D0 EE 8E 4A
41D8 03 20 AE 45 A2 00 A0 00
41E0 B1 FB DD 65 03 D0 0A C8
41E8 E8 EC 4A 03 D0 F2 20 68
41F0 41 20 1F 49 20 7C 41 B0
41F8 E3 20 2B 44 20 F0 40 90

4200 0D A0 2C 20 C4 40 20 6F
4208 42 20 E1 FF D0 EE 20 B6
4210 45 D0 8A 20 2D 49 20 F8
4218 47 20 38 49 20 C9 4D 48
4220 20 CF 42 68 20 E6 42 A2
4228 06 E0 03 D0 14 AC 4D 03
4230 F0 0F AD 58 03 C9 E8 B1
4238 FB B0 1D 20 65 42 88 D0
4240 F1 0E 58 03 90 0E BD E9
4248 4E 20 99 45 BD EF 4E F0
4250 03 20 99 45 CA D0 D2 60
4258 20 7B 42 AA E8 D0 01 C8
4260 98 20 65 42 8A 8E 4A 03
4268 20 FF 47 AE 4A 03 60 AD
4270 4D 03 20 7A 42 85 FB 84
4278 FC 60 38 A4 FC AA 10 01

4280 88 65 FB 90 01 C8 60 A8
4288 4A 90 0B 4A B0 17 C9 22
4290 F0 13 29 07 09 80 4A AA
4298 BD 98 4E B0 04 4A 4A 4A
42A0 4A 29 0F D0 04 A0 80 A9
42A8 00 AA BD DC 4E 8D 58 03
42B0 29 03 8D 4D 03 98 29 8F
42B8 AA 98 A0 03 E0 8A F0 0B
42C0 4A 90 08 4A 4A 09 20 88
42C8 D0 FA C8 88 D0 F2 60 B1
42D0 FB 20 65 42 A2 01 20 CE
42D8 40 CC 4D 03 C8 90 F0 A2
42E0 03 C0 03 90 F1 60 A8 B9
42E8 F6 4E 8D 54 03 B9 36 4F
42F0 8D 55 03 A9 00 A0 05 0E
42F8 55 03 2E 54 03 2A 88 D0

4300 F6 69 3F 20 D2 FF CA D0
4308 EA 4C 38 49 20 E6 47 A9
4310 03 20 9E 43 A0 2C 4C 3C
4318 45 20 F9 FD A9 3C 8D 13
4320 91 20 3A 43 A9 FF 8D 12
4328 91 A5 FB A0 18 20 34 43
4330 A5 FF A0 14 8D 10 91 8C
4338 11 91 A0 1C 8C 11 91 60
4340 20 54 48 85 FF 20 AE 45
4348 20 6E 41 20 7C 41 20 1C
4350 43 AD 49 03 0A 08 90 17
4358 A1 FB 8D 10 91 78 A9 C4
4360 8D 19 91 A9 3C 8D 11 91
4368 A9 20 2C 1D 91 F0 FB 20
4370 3A 43 58 8E 12 91 A9 0C
4378 8D 11 91 AD 10 91 28 B0

4380 04 10 02 81 FB C1 FB F0
4388 03 20 68 41 20 1F 49 D0
4390 B7 A9 4C 48 A9 77 48 08
```

```
4398 48 48 48 6C 60 03 8D 4B
43A0 03 48 20 8C 48 20 00 49
43A8 D0 F8 68 49 FF 4C 72 42
43B0 20 2B 44 AE 56 03 D0 0D
43B8 20 F0 40 90 08 20 C8 43
43C0 20 E1 FF D0 EE 4C 0E 42
43C8 20 AE 45 A2 2E A9 3A 20
43D0 0E 48 20 38 49 20 F8 47
43D8 A9 08 20 EA 48 A9 08 20
43E0 AB 43 20 38 49 20 38 49
43E8 A9 12 20 D2 FF A0 08 A2
43F0 00 A1 FB 29 7F C9 20 B0
43F8 02 A9 2E 20 D2 FF C9 22

4400 F0 04 C9 62 D0 06 20 E2
4408 4A 20 E5 4A 20 1F 49 88
4410 D0 DF 4C DF 4A 20 E6 47
4418 A9 08 20 9E 43 20 B6 45
4420 20 C8 43 A9 3A 8D 77 02
4428 4C 48 45 20 E6 47 85 FD
4430 86 FE 20 A4 49 F0 03 20
4438 EB 47 4C AE 45 20 31 48
4440 85 FD 86 FE A2 00 8E 66
4448 03 20 8C 48 C9 20 F0 F4
4450 9D 4F 03 E8 E0 03 D0 F1
4458 CA 30 14 BD 4F 03 38 E9
4460 3F A0 05 4A 6E 66 03 6E
4468 65 03 88 D0 F6 F0 E9 A2
4470 02 20 A4 49 F0 22 C9 3A
4478 F0 1E C9 20 F0 F3 20 90

4480 45 B0 0F 20 6C 48 A4 FB
4488 84 FC 85 FB A9 30 9D 65
4490 03 E8 9D 65 03 E8 D0 D9
4498 8E 54 03 A2 00 8E 56 03
44A0 A2 00 8E 4B 03 AD 56 03
44A8 20 87 42 AE 58 03 8E 55
44B0 03 AA BD 36 4F 20 70 45
44B8 BD F6 4E 20 70 45 A2 06
44C0 E0 03 D0 14 AC 4D 03 F0
44C8 0F AD 58 03 C9 E8 A9 30
44D0 B0 1E 20 6D 45 88 D0 F1
44D8 0E 58 03 90 0E BD E9 4E
44E0 20 70 45 BD EF 4E F0 03
44E8 20 70 45 CA D0 D2 F0 06
44F0 20 6D 45 20 6D 45 AD 54
44F8 03 CD 4B 03 D0 7F 20 21

4500 48 AC 4D 03 F0 2F AD 55
4508 03 C9 9D D0 20 20 F0 40
4510 90 01 88 C8 D0 6F 98 2A
4518 AE 53 03 E0 82 A8 D0 03
4520 B0 03 38 B0 60 CA CA 8A
4528 AC 4D 03 D0 03 B9 FC 00
4530 91 FB 88 D0 F8 AD 56 03
4538 91 FB A0 41 8C 77 02 20
4540 B6 45 20 C4 40 20 6F 42
4548 A9 20 8D 78 02 8D 7D 02
4550 A5 FC 20 9F 45 8E 79 02
4558 8D 7A 02 A5 FB 20 9F 45
4560 8E 7B 02 8D 7C 02 A9 07
4568 85 C6 4C 68 40 20 70 45
4570 8E 4A 03 AE 4B 03 DD 65
4578 03 F0 0D 68 68 EE 56 03

4580 F0 03 4C A0 44 4C 60 40
4588 E8 8E 4B 03 AE 4A 03 60
4590 C9 30 90 03 C9 47 60 38
4598 60 CD 4E 03 D0 1A 60 48
45A0 4A 4A 4A 4A 20 17 48 AA
45A8 68 29 0F 4C 17 48 A9 0D
45B0 20 D2 FF A9 0A 2C A9 91
45B8 4C D2 FF 8D 3F 03 08 68
45C0 29 EF 8D 3E 03 8E 40 03
45C8 8C 41 03 68 18 69 01 8D
45D0 3D 03 68 69 00 8D 3C 03
45D8 A9 80 8D 48 03 D0 26 A9
45E0 C0 8D 2E 91 A9 3F 8D 2E
45E8 91 20 94 48 D8 68 8D 41
45F0 03 68 8D 40 03 68 8D 3F
45F8 03 68 8D 3E 03 68 8D 3D

4600 03 68 8D 3C 03 AD 14 03
4608 8D 44 03 AD 15 03 8D 43
4610 03 BA 8E 42 03 58 AD 3E
4618 03 29 10 F0 03 4C 4E 40
4620 2C 48 03 50 1F AD 3C 03
4628 CD 5B 03 D0 6B AD 3D 03
4630 CD 5A 03 D0 63 AD 5E 03
4638 D0 5B AD 5F 03 D0 53 A9
4640 80 8D 48 03 30 12 4E 48
4648 03 90 D2 AE 42 03 9A A9
4650 45 48 A9 BA 48 4C 06 47
4658 20 AE 45 20 14 49 8D 4B
4660 03 A0 00 20 F2 48 AD 3D
4668 03 AE 3C 03 85 FB 86 FC
4670 20 38 49 A9 24 8D 4E 03
4678 20 16 42 20 E4 FF F0 FB

4680 C9 03 D0 03 4C 68 40 C9
4688 4A D0 4E A9 01 8D 48 03
4690 D0 47 CE 5F 03 CE 5E 03
4698 AD 21 91 C9 FE D0 3A A2
46A0 53 4C 5B 40 A9 00 F0 12
46A8 AD 5C 03 AE 5D 03 8D 5E
46B0 03 8E 5F 03 A9 40 D0 02
46B8 A9 80 8D 48 03 20 A4 49
46C0 F0 0F C9 20 D0 56 20 45
46C8 48 20 E3 48 20 A4 49 D0
46D0 4B 20 AE 45 AD 48 03 F0
46D8 1F 78 A9 A0 8D 2E 91 A9
46E0 5F 8D 2E 91 A9 DF A2 45
46E8 8D 44 03 8E 43 03 A9 49
46F0 A2 00 8D 28 91 8E 29 91
46F8 AE 42 03 9A 78 AD 44 03

4700 AE 43 03 20 98 48 AD 3C
4708 03 48 AD 3D 03 48 AD 3E
4710 03 48 AD 3F 03 AE 40 03
4718 AC 41 03 40 4C 60 40 20
4720 31 48 8D 5A 03 8E 5B 03
4728 A9 00 8D 5C 03 8D 5D 03
4730 20 42 48 8D 5C 03 8E 5D
4738 03 4C 68 40 20 CB 47 8D
4740 62 03 8E 63 03 20 42 48
4748 8D 4F 03 8E 50 03 20 42
4750 48 8D 51 03 8E 52 03 20
4758 A4 49 F0 0A 20 CF FF C9
4760 57 D0 03 EE 4E 03 20 21
4768 48 AE 56 03 D0 18 20 E7
4770 40 90 13 AC 4E 03 D0 1A
4778 B1 FB 20 87 42 AA BD F6

4780 4E D0 06 20 C4 40 4C 68
4788 40 AC 4D 03 C0 02 D0 33
4790 F0 03 8C 4D 03 88 38 B1
4798 FB AA ED 4F 03 C8 B1 FB
47A0 ED 50 03 90 1E 88 AD 51
47A8 03 F1 FB C8 AD 52 03 F1
47B0 FB 90 10 88 18 8A 6D 62
47B8 03 91 FB C8 B1 FB 6D 63
47C0 03 91 FB 20 1F 49 88 10
47C8 FA 30 9E 20 31 48 85 FD
47D0 86 FE 20 42 48 8D 54 03
47D8 8E 55 03 20 8C 48 20 45
47E0 48 85 FB 86 FC 60 20 31
47E8 48 B0 F6 20 45 48 B0 03
47F0 20 42 48 85 FD 86 FE 60
47F8 A5 FC 20 FF 47 A5 FB 48

4800 4A 4A 4A 4A 20 17 48 AA
4808 68 29 0F 20 17 48 48 8A
4810 20 D2 FF 68 4C D2 FF 18
4818 69 F6 90 02 69 06 69 3A
4820 60 A2 02 B5 FA 48 B5 FC
4828 95 FA 68 95 FC CA D0 F3
4830 60 A9 00 8D 59 03 20 8C
4838 48 C9 20 F0 F9 20 6C 48
4840 B0 08 20 8C 48 20 57 48
4848 90 07 AA 20 57 48 90 01
4850 60 4C 60 40 20 74 41 A9
4858 00 8D 59 03 20 8C 48 C9
4860 20 D0 09 20 8C 48 C9 20
4868 D0 0F 18 60 20 81 48 0A
4870 0A 0A 0A 8D 59 03 20 8C
4878 48 20 81 48 0D 59 03 38

4880 60 C9 3A 08 29 0F 28 90
4888 02 69 08 60 20 A4 49 D0
4890 FA 4C 65 40 A9 91 A2 43
4898 8D 14 03 8E 15 03 60 20
48A0 A4 49 F0 37 20 E6 47 A5
48A8 FB 05 FC F0 22 A5 9A C9
48B0 03 D0 9E A5 FB 8D 93 02
48B8 A5 FC 8D 94 02 A9 02 AA
48C0 A8 20 BA FF 20 C0 FF A2
48C8 02 20 C9 FF 4C 75 40 A9
48D0 02 20 C3 FF A9 03 85 9A
48D8 4C 68 40 A5 9A C9 03 F0
48E0 DC D0 F1 8D 3D 03 8E 3C
48E8 03 60 8D 4B 03 A0 00 20
48F0 38 49 B1 FB 20 FF 47 20
48F8 1F 49 CE 4B 03 D0 F0 60

4900 20 57 48 90 08 A2 00 81
4908 FB C1 FB D0 69 20 1F 49
4910 CE 4B 03 60 A9 3E 85 FB
4918 A9 03 85 FC A9 05 60 E6
4920 FB D0 09 E6 FF E6 FC D0
4928 03 EE 56 03 60 98 48 20
4930 AE 45 68 A2 2E 20 0E 48
4938 A9 20 4C D2 FF 20 0E 48
4940 A2 00 BD 76 4F 20 D2 FF
4948 E8 E0 1C D0 F5 A0 3B 20
4950 2D 49 AD 3C 03 20 FF 47
4958 AD 3D 03 20 FF 47 20 38
4960 49 AD 43 03 20 FF 47 AD
4968 44 03 20 FF 47 20 14 49
4970 20 EA 48 4C 68 40 4C 60
4978 40 20 31 48 20 E3 48 20

4980 42 48 8D 44 03 8E 43 03
4988 20 14 49 8D 4B 03 20 8C
4990 48 20 00 49 D0 F8 F0 DB
4998 20 CF FF C9 20 F0 F9 D0
49A0 06 20 F0 47 20 CF FF C9
49A8 0D 60 A0 01 84 BA A9 00
49B0 A2 65 A0 03 20 BD FF A8
49B8 20 E6 47 AD 49 03 C9 53
49C0 D0 08 20 A4 49 F0 AF 20
49C8 EB 47 20 98 49 F0 2D C9
49D0 22 D0 A3 20 CF FF C9 22
49D8 F0 0B 91 BB E6 B7 C8 C0
49E0 11 90 F0 B0 91 20 A4 49
```

```
49E8 FØ 12 2Ø 57 48 29 ØF FØ
49FØ 85 C9 Ø3 FØ 81 85 BA 2Ø
49F8 98 49 DØ D5 AD 49 Ø3 C9

4AØØ 53 DØ ØC A9 FB A6 FD A4
4AØ8 FE 2Ø D8 FF 4C 68 4Ø 49
4A1Ø 4C FØ Ø2 A9 Ø1 A6 FB A4
4A18 FC 2Ø D5 FF A5 9Ø 29 1Ø
4A2Ø FØ EA A9 69 AØ C3 2Ø 1E
4A28 CB 4C 6Ø 4Ø 2Ø E6 47 2Ø
4A3Ø A5 4Ø 4C 68 4Ø 2Ø E6 47
4A38 2Ø 1F 49 2Ø 1F 49 2Ø FØ
4A4Ø 47 2Ø 38 49 2Ø FØ 4Ø 9Ø
4A48 ØA 98 DØ 15 AD 53 Ø3 3Ø
4A5Ø 1Ø 1Ø Ø8 C8 DØ ØB AD 53
4A58 Ø3 1Ø Ø6 2Ø FF 47 4C 68
4A6Ø 4Ø 4C 6Ø 4Ø 2Ø E6 47 2Ø
4A68 7A 4A 4C 68 4Ø 2Ø AE 45
4A7Ø A2 2E A9 24 2Ø ØE 48 2Ø
4A78 F8 47 2Ø EA 4A 2Ø AØ 4A

4A8Ø 2Ø 38 49 2Ø 86 4A 2Ø 89
4A88 4A 2Ø 38 49 A2 Ø4 A9 3Ø
4A9Ø 18 ØE 54 Ø3 2E 55 Ø3 69
4A98 ØØ 2Ø D2 FF CA DØ EF 6Ø
4AAØ A5 FC A6 FB 8D 55 Ø3 8E
4AA8 54 Ø3 2Ø 38 49 A5 FC 2Ø
4ABØ B4 4A A5 FB AA 2Ø 38 49
4AB8 8A 29 7F C9 2Ø Ø8 BØ ØA
4ACØ A9 12 2Ø D2 FF 8A 18 69
4AC8 4Ø AA 8A 2Ø D2 FF C9 22
4ADØ FØ Ø4 C9 62 DØ Ø6 2Ø E2
4AD8 4A 2Ø E5 4A 28 BØ CØ A9
4AEØ 92 2C A9 14 2C A9 22 4C
4AE8 D2 FF 2Ø 38 49 A6 FB A5
4AFØ FC 4C CD DD 2Ø Ø5 4B BØ
4AF8 41 2Ø 38 49 2Ø F8 47 2Ø

4BØØ 7D 4A 4C 68 4Ø A2 Ø4 A9
4BØ8 ØØ 85 FC 2Ø C2 4B 2Ø 2B
4B1Ø 4B 85 FB 2Ø 22 4B 2Ø 3D
4B18 4B CA DØ F7 Ø8 2Ø 38 49
4B2Ø 28 6Ø 2Ø A4 49 FØ ØF C9
4B28 2Ø FØ ØB C9 3Ø 9Ø ØB C9
4B3Ø 3A BØ Ø7 29 ØF 6Ø 68 68
4B38 18 6Ø 4C 6Ø 4Ø 85 FE A5
4B4Ø FC 48 A5 FB 48 Ø6 FB 26
4B48 FC Ø6 FB 26 FC 68 65 FB
4B5Ø 85 FB 68 65 FC 85 FC Ø6
4B58 FB 26 FC A5 FE 65 FB 85
4B6Ø FB A9 ØØ 65 FC 85 FC 6Ø
4B68 2Ø C2 4B 8D 55 Ø3 48 48
4B7Ø 2Ø 38 49 2Ø 38 49 68 2Ø
4B78 FF 47 2Ø 38 49 68 AA A9

4B8Ø ØØ 2Ø F1 4A 2Ø 38 49 2Ø
4B88 86 4A 4C 68 4Ø 2Ø 9F 4B
4B9Ø 2Ø 38 49 2Ø F8 47 2Ø EA
4B98 4A 2Ø AØ 4A 4C 68 4Ø A2
4BAØ ØF A9 ØØ 85 FB 85 FC 2Ø
4BA8 C2 4B 2Ø 2B 4B 2Ø BC 4B
4BBØ 2Ø 22 4B 2Ø BC 4B CA DØ
4BB8 F7 4C 38 49 4A 26 FB 26
4BCØ FC 6Ø 2Ø 8C 48 C9 2Ø FØ
4BC8 F9 6Ø 2Ø 54 48 8D 88 Ø2
4BDØ A6 FB A4 FC 2Ø 8A FE A6
4BD8 FD A4 FE 2Ø 7B FE 2Ø 18
4BEØ E5 2Ø A4 E3 4C 68 4Ø 2Ø
4BE8 FØ 47 4C DB 47 2Ø E7 4B
4BFØ 18 A5 FB 65 FD 85 FB A5
4BF8 FC 65 FE 85 FC 4C ØD 4C
```

```
4CØØ 2Ø E7 4B 2Ø FØ 4Ø 84 FC
4CØ8 AD 53 Ø3 85 FB 2Ø 38 49
4C1Ø 2Ø F8 47 4C 68 4Ø A9 FØ
4C18 2C A9 ØØ 8D ØB 9Ø 4C 65
4C2Ø 4Ø 78 2Ø 52 FD 58 A9 3C
4C28 85 B2 AE 42 Ø3 9A A5 73
4C3Ø C9 E6 FØ 95 6C ØØ CØ 2Ø
4C38 E7 4B 2Ø 21 48 2Ø 38 49
4C4Ø AØ ØØ 8C 54 Ø3 8C 55 Ø3
4C48 2Ø FØ 4Ø 9Ø 1B AC 56 Ø3
4C5Ø DØ 16 18 B1 FB 6D 54 Ø3
4C58 8D 54 Ø3 98 6D 55 Ø3 8D
4C6Ø 55 Ø3 2Ø 1F 49 4C 48 4C
4C68 AD 55 Ø3 2Ø FF 47 AD 54
4C7Ø Ø3 2Ø FF 47 4C 68 4Ø AD
4C78 64 Ø3 DØ Ø4 A5 C6 DØ Ø3

4C8Ø 4C 56 FF AD 77 Ø2 C9 11
4C88 DØ 7D A5 D6 C9 16 DØ FØ
4C9Ø A5 D1 85 FD A5 D2 85 FE
4C98 A9 17 8D 5E Ø3 AØ Ø1 2Ø
4CAØ 51 4E C9 3A FØ 1A C9 2C
4CA8 FØ 16 C9 24 FØ 12 CE 5E
4CBØ Ø3 FØ CD 38 A5 FD E9 16
4CB8 85 FD BØ E1 C6 FE DØ DD
4CCØ 8D 49 Ø3 2Ø ØA 4E BØ B8
4CC8 AD 49 Ø3 C9 3A DØ 11 18
4CDØ A5 FB 69 Ø8 85 FB 9Ø Ø2
4CD8 E6 FC 2Ø C8 43 4C F4 4C
4CEØ C9 24 FØ 1A 2Ø C9 4D 2Ø
4CE8 6F 42 A9 ØØ 8D 4E Ø3 AØ
4CFØ 2C 2Ø 13 42 A9 ØØ 85 C6
4CF8 4C ØE 42 4C 56 FF 2Ø 1F

4DØØ 49 2Ø 6D 4A 4C F4 4C C9
4DØ8 91 DØ FØ A5 D6 DØ EC A5
4D1Ø D1 85 FD A5 D2 85 FE A9
4D18 17 8D 5E Ø3 AØ Ø1 2Ø 51
4D2Ø 4E C9 3A FØ 1A C9 2C FØ
4D28 16 C9 24 FØ 12 CE 5E Ø3
4D3Ø FØ 15 18 A5 FD 69 16 85
4D38 FD 9Ø E1 E6 FE DØ DD 8D
4D4Ø 49 Ø3 2Ø ØA 4E 9Ø Ø3 4C
4D48 56 FF AD 49 Ø3 C9 3A FØ
4D5Ø Ø6 C9 24 FØ 1D DØ 27 2Ø
4D58 DØ 4D 38 A5 FB E9 Ø8 85
4D6Ø FB BØ Ø2 C6 FC 2Ø CB 43
4D68 A9 ØØ 85 C6 2Ø Ø5 4E 4C
4D7Ø 7Ø 4Ø 2Ø DØ 4D 2Ø B2 4Ø
4D78 2Ø 7Ø 4A 4C 68 4D 2Ø DØ

4D8Ø 4D A5 FB A6 FC 85 FD 86
4D88 FE A9 1Ø 8D 5E Ø3 38 A5
4D9Ø FD ED 5E Ø3 85 FB A5 FE
4D98 E9 ØØ 85 FC 2Ø C9 4D 2Ø
4DAØ 6F 42 2Ø FØ 4Ø FØ Ø7 BØ
4DA8 F3 CE 5E Ø3 DØ EØ EE 4D
4DBØ Ø3 AD 4D Ø3 2Ø AB 43 A2
4DB8 ØØ A1 FB 8E 4E Ø3 A9 2C
4DCØ 2Ø 33 49 2Ø 16 42 4C 68
4DC8 4D A2 ØØ A1 FB 4C 87 42
4DDØ A6 D2 2Ø D7 4D A6 F4 E8
4DD8 86 AD 86 FE A2 ØØ 86 AC
4DEØ A9 2C 85 FD AØ CE E8 88
4DE8 B1 AC 91 FD 98 DØ F8 C6
4DFØ AD C6 FE CA 1Ø F1 A9 2Ø
4DF8 A6 D2 86 FE 84 FD AØ 2B

4EØØ 91 FD 88 1Ø FB A9 13 4C
4EØ8 D2 FF CØ 16 DØ Ø2 38 6Ø
4E1Ø 2Ø 51 4E C9 2Ø FØ F3 88
```

```
4E18 2Ø 3A 4E AA 2Ø 3A 4E 85
4E2Ø FB 86 FC A9 FF 8D 64 Ø3
4E28 85 CC A5 CF FØ ØA A5 CE
4E3Ø A4 D3 91 D1 A9 ØØ 85 CF
4E38 18 6Ø 2Ø 51 4E 2Ø 81 48
4E4Ø ØA ØA ØA ØA 8D 59 Ø3 2Ø
4E48 51 4E 2Ø 81 48 ØD 59 Ø3
4E5Ø 6Ø B1 FD C8 29 7F C9 2Ø
4E58 BØ Ø2 Ø9 4Ø 6Ø BD 98 4D
4E6Ø 2Ø D2 FF E8 DØ F7 6Ø ØØ
4E68 ØØ ØØ ØØ ØØ ØØ ØØ ØD 56
4E7Ø 49 43 32 3Ø 2Ø 4D 49 43
4E78 52 4F 4D 4F 4E 2Ø 56 31

4E8Ø 2E 3Ø 2Ø 2Ø 2Ø 42 49 4C
4E88 4C 2Ø 59 45 45 2Ø 32 32
4E9Ø 2Ø 4A 55 4E 45 2Ø 38 32
4E98 4Ø Ø2 45 Ø3 DØ Ø8 4Ø Ø9
4EAØ 3Ø 22 45 33 DØ Ø8 4Ø Ø9
4EA8 4Ø Ø2 45 33 DØ Ø8 4Ø Ø9
4EBØ 4Ø Ø2 45 B3 DØ Ø8 4Ø Ø9
4EB8 ØØ 22 44 33 DØ 8C 44 ØØ
4ECØ 11 22 44 33 DØ 8C 44 9A
4EC8 1Ø 22 44 33 DØ Ø8 4Ø Ø9
4EDØ 1Ø 22 44 33 DØ Ø8 4Ø Ø9
4ED8 62 13 78 A9 ØØ 21 81 82
4EEØ ØØ ØØ 59 4D 91 92 86 4A
4EE8 85 9D 2C 29 2C 23 28 24
4EFØ 59 ØØ 58 24 24 ØØ 1C 8A
4EF8 1C 23 5D 8B 1B A1 9D 8A

4FØØ 1D 23 9D 8B 1D A1 ØØ 29
4FØ8 19 AE 69 A8 19 23 24 53
4F1Ø 1B 23 24 53 19 A1 ØØ 1A
4F18 5B 5B A5 69 24 24 AE AE
4F2Ø A8 AD 29 ØØ 7C ØØ 15 9C
4F28 6D 9C A5 69 29 53 84 13
4F3Ø 34 11 A5 69 23 AØ D8 62
4F38 5A 48 26 62 94 88 54 44
4F4Ø C8 54 68 44 E8 94 ØØ B4
4F48 Ø8 84 74 B4 28 6E 74 F4
4F5Ø CC 4A 72 F2 A4 8A ØØ AA
4F58 A2 A2 74 74 74 72 44 68
4F6Ø B2 32 B2 ØØ 22 ØØ 1A 1A
4F68 26 26 72 72 88 C8 C4 CA
4F7Ø 26 48 44 44 A2 C8 ØD 2Ø
4F78 2Ø 2Ø 2Ø 5Ø 43 2Ø 2Ø 49

4F8Ø 52 51 2Ø 2Ø 53 52 2Ø 41
4F88 43 2Ø 58 52 2Ø 59 52 2Ø
4F9Ø 53 5Ø 41 42 43 44 46 47
4F98 48 4C 4D 4E 51 52 28 54
4FAØ 57 58 2C 3A 3B 24 23 22
4FA8 2B 2D 4F 49 4A 25 26 45
4FBØ 56 29 3D 5C FF AA 49 9F
4FB8 48 3D 44 1F 47 Ø2 41 F9
4FCØ 41 87 41 A4 46 AØ 41 AA
4FC8 49 BØ 43 3C 47 A8 46 4Ø
4FDØ 49 16 4C Ø6 41 B8 46 2A
4FD8 4C ØC 43 15 44 79 49 64
4FEØ 4A F4 4A 68 4B ED 4B ØØ
4FE8 4C 35 4A CA 4B 2C 4A 8D
4FFØ 4B 37 4C 21 4C AA 49 19
4FF8 4C 4Ø 43 4Ø 43 4Ø 43 E6
```

Three Notes On VIC Micromon

Using the VIC Micromon tape commands L, S, and V on a VIC-20 with 3K of RAM installed at $400 to $FFF will result in overwrite of $400 to $438 with file header characters (blanks). This is due to the tape buffer being relocated to $375 while in VIC Micromon from the normal $33C. The normal VIC cassette commands will work properly and not overwrite this area when you EXIT from VIC Micromon. This is because VIC Micromon restores the tape buffer pointer value to $33C when an EXIT is done. This problem does not occur if the 3K RAM at $400 to $FFF is not installed.

If the I (Initialize memory and screen pointers) command was used in VIC Micromon and you EXIT, then the RUN/STOP plus RESTORE should be used in addition to the NEW command to clean up the BASIC environment.

Any binary image saved on cassette tape with the VIC Micromon "S" command can be loaded in the normal VIC-20 BASIC environment by using the command: LOAD" ",1,1 which looks for the next program on tape and LOADs it into the same part of memory that it came from (see page 9 of VIC-20 *Programmer's Reference Guide*).

Checksum

There's a good amount of typing to do to enter the VIC Micromon program. Use the following BASIC program (after you've SAVEd a copy of your efforts) to locate any errors you might have made.

Program 3: VIC Micromon Checksum

```
10 DATA 12915,16867,15061,13732,14507,13829,1
   3801,12813,15027,13920
20 DATA 14355,11977,11877,13583,11338,15173,1
   2337,14852,14051,15723
30 DATA 13442,14047,14746,15059,13134,15848,1
   5858,17856,13327,8655
40 DATA 12171,10231
100 Q=16384
110 FORBLOCK=1TO32
120 FORBYTE=0TO127
130 X=PEEK(Q+BYTE):CK=CK+X
140 NEXTBYTE
150 READSUM
160 IFSUM<>CKTHENPRINT" ERROR IN BLOCK #"BLOCK
   :GOTO170
165 PRINT"                    BLOCK"BLOCK" I
   S CORRECT"
170 CK=0:Q=Q+128
180 NEXT BLOCK
```

©

For the Atari 800 with 810 disk drive, this is a quicker and simpler method of housecleaning diskettes.

Purge

Al Casper
Saukville, WI

It's summertime; free time for my Atari 800 is now carefully rationed. One of my favorite chores used to be clearing files off my diskettes, making room for new programs and files. Of course I'm kidding; I dreaded purging diskettes. First you had to load DOS and wait. File names had to be carefully entered, and finally the DELETE D:SLOW ? Y or N had to be dealt with. You also had to add one more step if the file was locked, or do it over from the start if you made a mistake. Repeat the above steps for each file you want deleted, and the entire process can easily take twenty minutes per diskette. Purge was written to make this job fast and easy, freeing your valuable time for other things.

Free Directory

When Purge is finished clearing your diskette, a directory is printed on the screen. The directory has two advantages over the DOS directory. First you do not need to load DOS to use it. Second, the files are printed in two columns, allowing twice as many files to be displayed before they start scrolling off the top of the screen.

The program is written in two short sections which makes it easy to save the "DIR" (Section A) as a separate program. The REMarks at the end of section A will explain this in more detail. I keep a copy of "DIR" on each of my diskettes. It requires only three sectors of disk space, well worth the time it can save you. I also have a LISTed version of "DIR" on file named "EDIR." I simply ENTER "D:EDIR" with any program I happen to have in memory. The high line numbers will almost never cause a conflict. Just type GOTO 32100 for a directory listing. "DIR" will now be a part of the program.

To use Purge, simply load the program, insert the diskette to be purged into disk drive one, and type RUN. One at a time the files on that diskette will be displayed on the screen. Pressing RETURN will display the next file. When an unwanted file is displayed, press CONTROL "P" to purge it. This process continues until all the files have been displayed. Don't panic if you make an error along the way; just press BREAK and start over. The purging takes place after all the files have been displayed and only if you press "P", as prompted on the screen. You'll hear a lot of action from the disk drive as the purging is taking place. The length of this operation varies with the number and length of files being deleted.

XIO Examples

The following is a line by line description of my program. This will be of most interest to programmers with limited experience working with disk operations. The XIO feature is the key to Purge. Writing this program in BASIC would have been very difficult without XIO. Note that the program listing does not have all the lines in correct order.

Line 32100 This special OPEN will allow inputs from the disk directory. The "*.*" in the file name is the same as a wildcard in DOS.

Line 32102 The TRAP is very useful. In this case it will detect the EOF (end of file), treat it as an error, and end the inputs.

Line 32104-32106 These are the INPUT(s) from the directory. The directory is printed in two columns.

Line 32110-32115 The file is CLOSEd, and the program goes into an endless loop to prevent possible information from scrolling off the screen.

Line 32000 Another TRAP for EOF. The keyboard (K:) is OPENed for input.

Line 32004 The OPEN is again to the directory.

Line 32006 One at a time each directory entry is INPUT and tested for "FREE SECTORS," which would be the last entry. The entry is then printed on the screen.

Line 32008 The program waits for an input from the keyboard. A chime sounds and slows things a bit.

Line 32010 If a purge was requested, the file name is created from the directory information.

Line 32012 The file name is saved in a larger string for later purging.

Line 32016-32017 Blank spaces have to be removed from the file name before they can be unlocked and deleted.

Line 32018 The XIO's perform unlock and delete just as if you were using DOS.

Line 32030 Files are CLOSEd, and the "DIR" routine will follow.

Program 1: Section A: DIR

```
32050 REM SECTION  (A)  DIR
32055 REM
32060 REM WHEN FINISHED TYPING THIS S
      ECTION SAVE IT WITH THE FILE NA
      ME  'D:DIR'.
32065 REM ALSO LIST IT TO THE DISKETT
      E WITH THE FILE NAME   'D:EDIR'
      .
32067 REM 'EDIR' CAN THEN BE ENTERED
      AT ANY TIME TO ATTACH A 'DIR'
32068 REM TO YOUR PROGRAM TO BE CALLE
      D WITH A 'GOTO 32100'.
32070 REM THEN CONTINUE ADDING SECTIO
      N (B)  TO SECTION (A)
32100 OPEN #5,6,0,"D:*.*"
32102 CLR :GRAPHICS 0:POKE 82,1:DIM E
      NT$(17):TRAP 32110:? :? "  DISK
       DIRECTORY
       "
32104 INPUT #5,ENT$:? ENT$;"     ";
32106 INPUT #5,ENT$:? ENT$:GOTO 32104
32110 CLOSE #5:? :? "    END       P
      RESS BREAK     END   ";:POKE 82
      ,2
32115 GOTO 32115
```

Program 2: Section B: Purge

```
31900 REM SECTION  (B)  PURGE
31910 REM
32000 TRAP 32013:OPEN #4,4,0,"K:":DIM
       E$(17),S$(500),PG$(14):X=1:Y=14
32002 GRAPHICS 0:? "{DOWN}TO PURGE":?
       "{DOWN}AFTER EACH FILE DISPLAY
      ED PRESS":? "CONTROL-P TO DELET
      E OR   PRESS RETURN"
32004 ? "TO CONTINUE":OPEN #5,6,0,"D:
      *.*"
32006 INPUT #5,E$:POSITION 2,10:IF E$
      (5,16)<>"FREE SECTORS" THEN ? E
      $:? :? " CHOICE";:GOTO 32008
32007 GOTO 32013
32008 GET #4,K:IF K<>16 THEN POSITION
       2,12:? " CHOICE   ":FOR Q=15 TO
       0 STEP -0.2:SOUND 0,20,10,Q:NE
      XT Q:GOTO 32006
32010 PG$(1,2)="D:":PG$(3,10)=E$(3,10
      ):PG$(11,11)=".":PG$(12,14)=E$(
      11,13)
32012 S$(X,Y)=PG$:X=X+14:Y=Y+14:FOR Q
      =15 TO 0 STEP -0.2:SOUND 0,40,1
      0,Q:NEXT Q:GOTO 32006
32013 POSITION 2,15:? "PRESS P TO PUR
      GE";:FOR Q=1 TO 120:POKE 764,25
      5:NEXT Q:GET #4,K:IF K=80 THEN
      32015
32014 GOTO 32020
32015 X=1:Y=14:S=0
32016 TRAP 32020:PG$=S$(X,Y):FOR Q=1
      TO 13:S=S+1:IF PG$(S,S)=" " THE
      N PG$(S,14)=PG$(S+1,14):S=S-1
32017 NEXT Q
32018 XIO 36,#3,0,0,PG$:XIO 33,#3,0,0
      ,PG$:X=X+14:Y=Y+14:S=0:GOTO 320
      16
32020 CLOSE #4:CLOSE #5
```

©

Written for the Apple II and Apple II Plus, this article explains the sounds of the Apple, with special attention to its Music routine.

Apple Sounds

Michael P. Antonovich
Wyomissing, PA

When you are writing software for the Apple II, are you afraid to add sound to your program because you feel that it is too hard or that you have a tin ear? I have tested many Apple programs, and those which incorporate sound, quality graphics, and user-friendly input are often the ones which stand out in my mind. If you haven't been using sound, read on and see how easy Apple sounds really are.

When you bought your Apple II, you received a chip called the Programmer's Aid already plugged into your motherboard. This chip (a ROM – Read Only Memory) contains a group of utilities for the Integer BASIC program. Some of you probably have the Apple II Plus rather than an Apple II. However, don't stop reading if you bought the Language Card.

Maybe you have never looked at what the Programmer's Aid does. If you have, you noticed a utility called the *Music* routine. The Music routine lets you create music with the options of changing the note's pitch, duration, and timbre.

To play a note, you have to POKE three items into memory before you can call the Music routine. The first item is the timbre. Timbre will make the same note sound slightly different, but you have only five possible timbre selections. They are: 2, 8, 16, 32, and 64. The timbre you want must be POKEd into decimal memory location 765 as follows:

POKE 765,32

Timbre 32 will give you the cleanest notes over the Apple's range.

The second item you need to store is the note's duration. A decimal value from 1 to 255 can be POKEd into decimal address 766 to produce short to long notes, respectively. A value of 170 will result in a note of approximately one-second duration. The POKE command is:

POKE 766,170

Third, you must select the note which you want. The note decimal values can range from 1 to 50, with 1 being the low end of the scale. The numbers are based on a chromatic scale. Values of the notes above 50 will result in a random note selection. Middle C corresponds roughly to the POKE:

POKE 767,32

Finally, you are ready to play the note. To do this, you must call the machine language routine located at decimal address -10473 or 55063, using a CALL statement as in:

CALL -10473

There! How did that sound? Just one note, you say? Well, the following program lists about a dozen sounds made using the above techniques. They are by no means the limits of the Apple's ability. Rather, they are presented to whet your appetite so that you will try to create some sounds on your own.

```
9 REM  FALLING SOUND
10 POKE 766,2: POKE 765,32: FOR I=50 TO 1 STE
    P -1: POKE 767,I: CALL-10473:NEXTI
11 END
19 REM RISING SOUND
20 POKE 766,2: POKE 765,32: FOR I=1 TO 50: PO
    KE 767,I: CALL -10473: NEXT I
21 END
29 REM VARIOUS WHIRLING SOUNDS
30 POKE 766,2: POKE 765,32: FOR I=20 TO 30: P
    OKE 767,I: CALL -10473: NEXT I
31 FOR I=50 TO 10 STEP -1: POKE 767,I: CALL -
    10473: NEXT I: GOTO 30
40 POKE 766,2: POKE 765,32: FOR I=20 TO 30: P
    OKE 767,I: CALL -10473: NEXT I
41 FOR I=30 TO 20 STEP -1: POKE 767,I: CALL -
    10473: NEXT I: GOTO 40
50 POKE 766,2: POKE 765,32: FOR I=10 TO 40: P
    OKE 767,I: CALL -10473: NEXT I
51 FOR I=30 TO 20 STEP -1: POKE 767,I: CALL -
    10473: NEXT I: GOTO 50
59 REM WARNING SIREN
60 POKE 766,2: POKE 765,32: FOR I=1 TO 100: P
    OKE 767,40: CALL -10473: NEXT I
61 POKE 766,4: FOR I=30 TO 20 STEP -1: POKE 7
    67,I: CALL -10473: NEXT I
62 FOR I=10 TO 40: POKE 767,I: CALL -10473 NE
    XT I: GOTO 60
64 REM LIGHT SABER
65 POKE 766,2: POKE 765,32: FOR I=1 TO 100: P
    OKE 767,1: CALL -10473: NEXT I
66 FOR I=10 TO 40: POKE 767,I: CALL -10473: N
    EXT I
67 FOR I=30 TO 20 STEP -1: POKE 767,I: CALL -
    10473:  NEXT I: GOTO 65
69 REM   PADDLE 0 CONTROLLED MOTOR
70 POKE 766,2:POKE 765,32:POKE 767,140:CALL-1
    0473:FOR I=1 TO PDL(0):NEXTI:GOTO70
79 REM  PADDLE CONTROLLED SOUNDS
80 POKE 766, PDL (1): POKE 765,32: POKE 767, ~
    PDL (0): CALL -10473: GOTO 80
89 REM  RANDOM NOISE
90 D=64
91 POKE 766,2:POKE 765,D:POKE 767,100:CALL -1
    0473:D=D/2:IF D<1 THEN D=64:GOTO91
99 REM  ALIEN SPACESHIP SOUNDS
100 D=64
101 POKE 766,2:POKE 765,D:POKE 767,150:CALL-10
    473:D=D/2:IF D<1THEN D=64:GOTO101
110 D=2
111 POKE 766,2:POKE 765,D:POKE 767,150:CALL-10
    473:D=D*2:IF D>64 THEND=2:GOTO111
199 REM  MORE SOUNDS
210 POKE 766,2:POKE 765,32:FORI=50TO1STEP-1:PO
    KE767,I:CALL-10473:NEXT I:GOTO210
220 POKE 766,2:POKE 765,32:FOR I=1 TO 50:POKE ~
    767,I:CALL -10473:NEXT I:GOTO 220
999 END
```

©

A quick way to verify cassettes, a survey of languages available for the Atari, and a fix for a bug in Atari's RS-232 handlers.

Insight: Atari

Bill Wilkinson
Optimized Systems Software
Cupertino, CA

Well, I didn't quite make it. I was trying to have a cassette verify program done in time for this month's column, but the pressures of writing a couple of sections for the new *COMPUTE!'s Book of Atari Graphics,* producing three major new OSS products, and answering literally hundreds of phone calls got to me. So, wait for next month. But in the meantime, at least I have a quickie verify method that might keep the frustrations away for a month.

Quick And Dirty

One of the major flaws of the Atari computers has always been the lack of a cassette verify capability. But there is an almost effortless way to simulate this missing capability.

The secret lies in the fact that, because of Atari's superior operating system and because BASIC interfaces properly to it, you can LIST to any file *or device.* So, when you are ready to save your program to cassette, do *not* use CSAVE. Instead, Use LIST"C:" to produce an ATASCII listing on the cassette. Then you can rewind the tape and, *without deleting or changing the program in memory,* enter the following direct statements:

```
DIM Q$(256) : OPEN #1,4,0,"C:"
FOR Q= TO 100000:INPUT #1,Q$:PRINT Q$:
NEXT Q
```

Do you see the reason for the trick? Atari makes no distinction between a listing file and a data file, even on a cassette, so we can simply read the listing as data and print what we read on the screen. If what appears on the screen is correct, the cassette was recorded correctly. Incidentally, the FOR/NEXT loop is only needed so that we can enter the statements in direct mode (without line numbers). The loop will execute more times than there are lines in the file, but the end of file error will stop the process anyway. (And it is a good idea to type "END" after getting the end of file error.)

For the adventuresome, there might be an even easier way. After using the LIST"C:", simply rewind the tape and type ENTER"C:". Remember, ENTER does *not* erase the program in memory, but instead merges the filed program with the current one. But in this case, since the two programs are the same (if the file was recorded correctly), the ENTER should have no visible effect. If there is an error in the tape, the ENTER will simply halt and no harm will be done. Theoretically. In truth, it is possible that one line could be destroyed (if it were partially ENTERed from one tape block and then blew up in the next block). I have not tested this exhaustively, so use it at your own risk.

Foreign Languages

What is *the* Atari language? What is the best language for doing the most things with an Atari computer? Is there such a thing? There may be no good answers to these questions, but trying to answer them may prove interesting, so let's give it a shot.

The Atari now has a respectable complement of languages available for it. I will list those I know of here and I must apologize in advance for any omissions. The listings within each category are roughly in order of date of introduction of the product. An asterisk indicates a product no longer actively advertised, so check with the publisher for availability.

Assemblers

- Cassette-Based Assembler – Quality Software*
- Assembler/Editor Cartridge – Atari, Inc.
- EASMD (Edit/ASseMble/Debug) – OSS, Inc.*
- DATASM/65 – Datasoft*

Macro Assemblers

- MAE (Macro Assembler/Editor) – Eastern House
- Macro Assembler – ELCOMP
- AMAC (Atari MACro assembler) – Atari, Inc.
- MAC/65 – OSS, Inc.

Interpreters

- Atari BASIC – Atari, Inc.
- BASIC A# – OSS, Inc.
- LISP – Datasoft
- PILOT – Atari, Inc.
- tiny c – OSS, Inc.
- Microsoft BASIC – Atari, Inc.

Pseudo-Compilers

- QS FORTH – Quality Software
- Atari FORTH – Atari Program Exchange
- pns FORTH – Pink Noise Studios
- PASCAL – Atari Program Exchange
- ValForth – Valpar

Compilers

- PASCAL – Atari Program Exchange
- C/65 – OSS, Inc.

I admit I hesitated over classifying FORTH as a pseudo-compiled language, but I was trying to

GEMINI–
FOR PRINTER VALUE THAT'S OUT OF THIS WORLD

Over thirty years of down-to-earth experience as a precision parts manufacturer has enabled Star to produce the Gemini series of dot matrix printers—a stellar combination of printer quality, flexibility, and reliability. And for a list price of nearly 25% less than the best selling competitor.

The Gemini 10 has a 10″ carriage and the Gemini 15 a 15½″ carriage. Plus, the Gemini 15 has the added capability of a bottom paper feed. In both models, Gemini quality means a print speed of 100 cps, high-resolution bit image and block graphics, and extra fast forms feed.

Gemini's flexibility is embodied in its diverse specialized printing capabilities such as super/sub script, underlining, backspacing, double strike mode and emphasized print mode. Another extraordinary standard feature is a 4k buffer (with an additional 4k on the serial board). That's twice the memory of leading, comparable printers. And Gemini is compatible with most software packages that support the leading printers.

Gemini reliability is more than just a promise. It's as concrete as a 180 day warranty (90 days for ribbon and print head), a mean time between failure rate of 5 million lines, a print head life of more than 1 million characters, and a 100% duty cycle that allows the Gemini to print continuously. Plus, prompt, nationwide service is readily available.

So if you're looking for an incredibly high-quality, low-cost printer that's out of this world, look to the manufacturer with its feet on the ground—Star and the Gemini 10, Gemini 15 dot matrix printers.

star®
MICRONICS • INC

MAKING A NAME FOR OURSELVES

1120 Empire Central Place, Suite 216, Dallas, TX 75247
For more information, please call Bob Hazzard, Vice President, at (214) 631-3560.

group the products by speed and space considerations. Technically, FORTH is a "threaded" language, but that doesn't imply anything about its implementation. Besides, I love to bug the FORTH aficionados. Anyway, to proceed.

The assemblers grow more numerous almost monthly, and it is obvious that most serious graphics work for the Atari is still being done in assembly language, even though the 6502 has one of the strangest assembly languages in existence. (There used to be others far stranger, but they've either died out or been relegated to the dedicated controller market. You know you're an old-timer if you ever used a 4004, PPS-4, PPS-8, 8008, F-8, 2650, COPS, TI1000, etc.)

Of course, Macro Assemblers are a step in the right direction, but I have yet to see any 6502 assembler system done "right," with relocatable *and* linkable object modules, a symbolic debugger, and more. Yet. For those of you not familiar with macro-assembly techniques, I should point out that old macro hackers usually build up a library of their favorite macros and can easily plug together several variations on a utility program (for example) by simply picking and choosing from their assortment of macros.

I don't really want to explore this subject in depth right now, but I would like to point out that, using some – or at least one – of the currently available macro assemblers for the Atari, you can write assembly language programs that look like this:

```
    OPEN   1,8,0,"D:NEWFILE"
LOOP
    INPUT   1,LINE
    IFERROR   EXIT
    PRINT   0,LINE
    GOTO   Loop
;
EXIT
    ...
```

It would seem to me that the percentage of Atari owners who will successfully dive into assembly language is too small to make *any* assembler become the dominant Atari language. Currently, though, there is no other way to write such marvels as Eastern Front, Frogger, and operating systems. So, at least for many software heavyweights, assembly is *the* language.

Compiling 6502 Code

I'd like to skip the interpreters for now and discuss both kinds of compilers. For starters, what's the difference between a compiler and a pseudo-compiler? Software purists could argue this point for days, but I will use a simple rule here: if it produces output, it's a compiler. If it produces tokens or words which must be interpreted, it's a pseudo-compiler.

Now, quite honestly, on a 6502 there probably isn't much advantage in one of these over the other. Generally, a pseudo-compiler produces fewer bytes of code, but requires a relatively massive runtime support module (the interpreter, including I/O routines, etc.). As a rule, on most computers, pseudo-compiled code will run slower than compiled code because of the overhead of the interpreter.

Unfortunately, most conventional language compilers for 6502-based machines will of necessity produce large and generally clumsy code. Consider the following statement, legal, with minor variation, in most higher level languages:

```
array(index) = value ;
```

Given that all three variables shown are 16-bit globals, a *really* good compiler for a Z80 could produce as few as 15 bytes of code to execute it (and the one we wrote for Cromemco produces only 16 bytes).

A *superb* compiler for the 6502 could produce as few as 25 bytes, but only if it *knew* that "index" would not contain a value exceeding 127! And, oh yes, most pseudo-code compilers would probably produce 11 or 12 bytes of tokens for this same code.

So, you see, even a multi-pass optimizing compiler can *at best* coax the 6502 into using 1.5 to 2.5 times the amount of code that a Z80 needs. And, in truth, there aren't any "superb," "multi-pass," "optimizing" compilers yet available for the Atari. So the code generated will be even bigger, perhaps as much as three to four times that needed by a Z80. (To be fair, an "average" Z80 compiler would produce 25 or so bytes of code, itself.)

So why did we digress through all of this? Simply to show that it is remarkable that there are any compilers at all for the Atari. Of the two compilers shown, the PASCAL is the more complete language, but it is a little difficult to work with, needs a huge support library, and requires two disk drives. Still, since it is an APX product, it is a remarkable bargain. C/65, on the other hand, is a subset of the full C language; it is a one-pass compiler (no optimizing here, obviously) which produces macro assembly language output. Its primary advantages: the assembly language can produce a listing with the original C code interspersed as comments, it uses a very small support library, and it can run on a single drive. But I think we may not have seen the end of compiler efforts on the 6502.

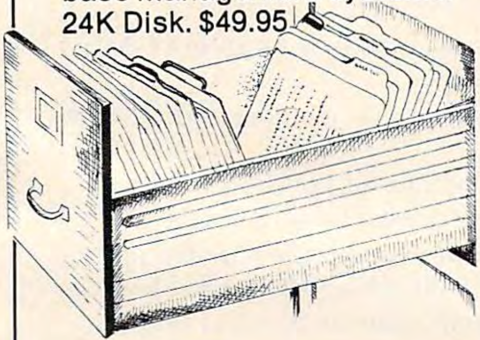
Interpreter Efficiency

But now we come to my favorite topic: interpreters. Despite its shortcomings as a compiled-for machine, the 6502 comports itself nicely when interpreting:

SWIFTWARE®

FILE-IT 2 by Jerry White

A powerful financial database management system. 6 user defined fields are created with up to 5 sub-fields beneath each main field. Alphabetically handles data and also does math computation on any selected fields. Data files are stored on separate disks with full field and sub-field sorting with file merging. Supports up to 4 drives including the 128K Axlon Ramdisk. The "alternative" to more costly database management systems. 24K Disk. $49.95



DATALINK by Tony Dobre

Top rated by national magazine reviews, purchased by NASA, this ultra-sophisticated menu-driven multioption smart terminal communications package supports uploading/downloading in full-duplex or simplex modes. Compatable with all the commercial services and bulletin boards such as the Source and Compuserve, etc.

24K Disk. $39.95

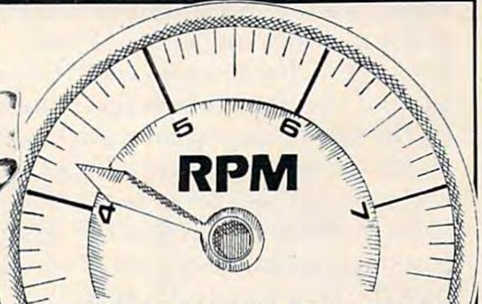
MUSIC LESSONS

Jerry White's MUSIC LESSONS

MUSIC LESSONS has everything you need to know to create your own beautiful music and a wide range of sound effects on your ATARI computer. This comprehensive tutorial contains 13 separate programs and PLAYER PIANO on 2 cassettes or 1 double sided disk and includes extensive documentation complete with program listings. 32K DISK 16K CASSETTE (24K required for PLAYER PIANO). $29.95

SPACE SHUTTLE by Paul Kindl

Join the crew of the *Space Shuttle* as they prepare to take the next step into the world of space travel. Take control of the world's first reuseable spaceship, the *Space Shuttle*, and in an accurate full graphic simulation, place yourself in the cockpit. Pilot the *Space Shuttle* through take-off with booster stage separations, orbit, descent down the glide path and landing to touchdown — complete with a chase plane and scrolling runway visible through the cockpit windscreen. You assume command throughout all phases of the mission aided by complete instrumentation. 32K Diskette $29.95.



TACHMASTER by Tony Dobre

This highly accurate Disk Drive speed diagnostic utility program has been designed specifically for the Atari 810 Disk Drive by the author of the original *"RPM"* program. *TACHMASTER,* using 100 percent machine language routines, displays RPM readings, updated five time per second, in digital and analog form on your screen. Reliability to within one quarter RPM is assured. With *TACHMASTER,* Drive speed adjustment to factory specifications can be easily accomplished without any special equipment. Comes complete with step by step user's instruction manual. Disk Only. $29.95.

AVAILABLE AT SELECT COMPUTER STORES

MAIL ORDERS: Send check or money order plus $2.50 shipping and handling. N.Y. Residents add 7¼% sales tax.

TELEPHONE ORDERS:
(516) 549-9141

VISA MasterCard

SWIFTY SOFTWARE, INC.
64 Broad Hollow Road
Melville, New York 11747

Send for FREE catalog

Dealer Inquires Invited

it is fast and needs only relatively compact code to implement. Why? Simply because interpreters generally work on "lines" of input. But if we limit a line to 256 characters (a very reasonable limitation), we find that there are several modes of operation on the 6502 that just love working with such short character strings. (Especially, of course, the "indirect indexed" or "(zero page),Y" instructions.) The truth of the matter is that the designer of a 6502-based interpreter has a lot of leeway in prescribing how the language will run best.

So look at the wealth of interpreters available already! With more to come, I am sure. We find in these interpreters the most used of all Atari languages, Atari BASIC. Well, that's not surprising, considering that it's essentially a required ingredient in an Atari system. But let's come back to it in a moment.

Naturally, PILOT is here. It's a nice, simple language which can easily be interpreted. It was probably a joy to program; I would have loved being involved.

But there are some real powerhouse languages here, also. LISP has traditionally been an interpreter, the darling of the Artificial Intelligence people. And, finally, there is Microsoft BASIC and BASIC A+. Quite honestly, I feel that these last two languages provide the best and easiest access to the Atari's features. Naturally, I am prejudiced towards BASIC A+, but the Microsoft BASIC has a few nice and unique features even if it isn't quite as easy to use.

What's *The* Atari Language?

So, after all that, just what is *the* Atari language? Well, I'm going to cop out and say that it's Atari BASIC. Despite all the nasty things said about the poor thing, look at all the things written in BASIC. And they work.

Atari BASIC is an excellent starting point. The easiest next step is BASIC A+, but most people won't have too much trouble learning other algebraic languages, such as PASCAL or C (the only real problem with these languages is that debugging is so much harder than with an interpreter). I consider PILOT and LISP useful languages in their own right, but much of what you learn in them is non-transportable to other languages.

The same is true of FORTH. FORTH enthusiasts would have you believe that FORTH is the only language you will ever need. Nonsense. Each language has its uses, its strong points, and its failings. (In my opinion, the major failings of FORTH are (1) that it operates independent of the host system's DOS and (2) techniques learned in FORTH are often non-transportable to other languages, because of FORTH's reverse-Polish notation. However, I respect the language for what it is: a hacker's dream come true. And I'm a hacker.)

Personally, I like to collect languages the way other people collect games. Seldom will I find one that won't teach me something new about how computers can be made to work. So try some "foreign" languages yourself soon and see how much fun they can be. (And pain and trouble and frustrating and educational and uplifting.)

System Reset And The 850

A couple of times in the past, I have presented in this column the "rules" for adding device drivers to Atari OS. Well, would you believe it, Atari itself broke the rules when they implemented the 850 (RS-232) handlers. The violation was a minor one, yet the consequences *can* be severe. To start with, let's recap my rules:

1. Locate the current value of system LOMEM (contents of $02E7).
2. Load your driver into memory and relocate it to LOMEM.
3. Adjust the contents of LOMEM to reflect the memory being used by your driver.
4. Add your device's name and handler address to the handler table (HATABS, at $031A).
5. Get the current value of DOSINI (location $000C) and save it somewhere in your handler. Put your own initialization address into DOSINI.
6. Whenever your initialization routine is called (i.e., when System Reset is pushed by a user), first call the initializer whose address was in DOSINI before you changed it. Then perform steps 3 and 4 again, since Reset will have changed LOMEM and reloaded the HATABS.

Now step 2 is the most difficult of these to accomplish, in practice, because it is hard to produce a relocatable module on the Atari. Many programs I have seen (and written) are actually assembled absolute at a "known" good location. This is okay, if you are writing for your own private system: you know what will be loaded when and where. But if you are producing a driver for sale, you really should follow the rule faithfully.

Atari's 850 drivers do, indeed, relocate themselves beautifully. They add their name to the handler table. They adjust the system LOMEM pointer. So what do they do wrong? One minor thing: they do steps 3 and 4 *before* they call the old initialization routine (see step 6) instead of *after*!

The result: the 850 handler changes LOMEM to just above itself and then calls the DOS initialization, which resets LOMEM to just above DOS!


PCA
PRESENTS
LABYRINTHS
FOR ATARI® 400/800
A
MAGICAL,
MYSTICAL,
AND THOROUGHLY ENTERTAINING WORLD OF
MONSTERS, DEMONS & TREASURES
THAT IS COMPLETELY
DIFFERENT EVERYTIME YOU PLAY!
YOU WILL START AS A FIGHTER, THIEF, CLERIC OR MAGIC-USER.
TRY TO GAIN EXPERIENCE & POWER TO ADVANCE TO
HIGHER LEVELS OF POWER.
Discover GOLD, WEAPONS, ARMOR, and MAGIC SCROLLS among the
TREASURES of those you defeat. But BEWARE...
Your power leads you to more deadly monsters who may have magic powers!
A wide choice of COMMANDS combined with SOUND EFFECTS
makes for unparalleled computer melee!
OVER 35 completely different types of monsters
.... LABYRINTHS ....
CASSETTE - $24.95
(32K required)
DISK - $28.95
(48K required)
Progressive Computer Applications
is setting new standards of quality in microcomputer software.
In this tradition, we are also presenting
another supreme simulation ...
COMPUTER EXCHANGE
An OUTSTANDING STOCK MARKET simulation
Solitaire, or up to 4 players trade stocks in
a fictional stock market.
One joy stick required per player
.... COMPUTER EXCHANGE ....
CASSETTE - $21.95
(16K required)
DISK - $24.95
(32K required)
ORDER FROM:
P.C.A.
P.O. BOX 46
BURTONSVILLE, MD. 20866
Dealer inquires welcome

Thus, the RS-232 handlers are not protected from programs which come in and *quite properly* use RAM starting at LOMEM. Generally, if you are running with Atari BASIC, this won't affect you, since BASIC maintains its own pointer to LOMEM once it is initialized at power on. But if you return to DOS without MEM.SAV, or run some assembly language utility... well, there are just too many cases where this little *faux pas* can wipe you out.

I am currently working on a patch (ready by next month, I hope) to the handler (to be made via the handler loader) which will fix this problem. In the meantime, it might be a good idea to have your programs check for the existence of the "R" name in HATABS and avoid the appropriate amount of memory if it is found.

In December we'll have some heavy assembly language stuff, what with the patch to the 850 handler and the cassette verify routine. I hope to return to some more BASIC stuff to start off the new year. ©

CAPUTE!

Modifications Or Corrections To Previous Articles

TI 99/4A Charades

Our thanks to Steve Davis, author of "Charades" for the TI 99-4A (September 1982, p. 64), for pointing out the following typos in the program listing:

```
831 DATA SHARP AS A TACK
1220 CALL CLEAR
1330 IF STATUS=0 THEN 1340 ELSE 1350
1370 RETURN
1580 CALL SOUND
```

PET Machine Language Compactor

Author David Evans has provided some readers with a faster version of his "Compactor" (July 1982, p. 159). To make all versions work correctly, he suggests that the following line be typed in and then the corrected version be saved via the monitor (the start and end addresses are \$0363, \$0B78):

```
IF PEEK (2461) = 12 THEN POKE 2461,13
```

©

Here's an expert's explanation of how to telecommunicate with the help of machine language.

A Monthly Column

Machine Language:

Serial Communications

Jim Butterfield
Associate Editor

So you want to communicate? If you want to reach another device or computer over a distance of 20 feet or so, you can string eight wires or more and spit out all eight bits of a byte at one time. But to go across town or around the world, you'll have to send one bit at a time, one behind the other. That's serial transmission.

We'll take a small part of the communications interface – the part that changes the computer's bytes into serial and back – and discuss the machine language approach.

What Is It?

Most small computers run their communications at a modest speed, say 300 bits per second. (Don't say "baud": it has a special meaning in telecommunications and will just get things muddled.) At this low speed, we use a type of transmission called "asynchronous" (a-SINK-ron-uss). In non-technical terms, this means send a character when you feel like it, and send nothing if you choose. This makes it easy for the sender: if there's nothing to send, don't worry about it. It's a little tougher for the receiver, which must keep an eye on the line and decide if there's a character coming in or not.

The usual code used for communications characters is ASCII. Some books tell you that it's a seven-bit code. Don't believe it. For communications purposes, we send eight bits of data. Wait, there's more. We have to send a little extra because of this asynchronous thing.

Before we send the eight bits, one behind the other, we must send a "start bit." This tells the other end, "Look out, there's a character coming!" After we have sent the start bit plus the eight data bits, we enforce a quiet time of one-bit length. This helps the receiving end catch up in case we're getting a bit ahead. This quiet time is called the "stop bit," and it's a minimum wait time only. If we don't have another character to send, the quiet time continues far beyond a single-bit time.

The receive end must spot the start bit, and then start clocking in the eight bits of data that follow it. It can ignore the stop bit pause at the end, but might choose to check that it's there so as to spot possible errors.

Setting The Quiet Line

The outgoing line is often one of the output ports. We'll need to separate it from the other ports. To set the line to the quiet state (called "marking"), we'll probably want to set the port to a binary one. Let's assume that we're using bit six of an I/O register at E84F. We'd code: LDA $E84F:ORA #$40:STA $E84F and the line would now be set to the idle, marking state.

Starting The Character

Suddenly, we have a character to send. We'll store it into location CHAR, and set a value of 8 into location COUNT to count the bits as they go out. Now our job is to send the start bit. That's a zero, or spacing, signal. But wait! We must decide about the timing.

Three hundred bits per second works out to a timing of 3,333 microseconds per bit. That's the value we must place in our timer if it counts in microseconds: 3333 or hexadecimal 0D05. The value might need to be slightly adjusted, but it's close. We might code a subroutine:

```
SPACE LAD $E84F
      AND #$BF   (clear out bit)
      STA $E34F
TIMER LDA #$05
      STA $E848  (timer low)
      LDA #$0D
      STA $E849  (timer high)
WAIT  BIT $E849  (watch timer)
      BPL WAIT
      RTS
```

This routine will hang in a stall loop until the time is up. It seems inefficient – interrupts could do the job more efficiently – but maybe we have nothing better to do anyway until the time has gone by.

Sending The Bits

Our start bit has gone. Now for the eight bits of our byte. Let's count down with DEC COUNT, and, if all the bits have gone, we can exit with BEQ EXIT. Otherwise, we get the bit (low order first) with LSR CHAR. The bit we want pops into the Carry flag. If the bit is zero (carry clear), we want to call subroutine SPACE again. If the bit is one (carry set), we must call a new subroutine, MARK:

```
MARK    LDA $E84F
        AND #$40
        STA $E84F
        BNE TIMER
```

The last branch is unconditional; the AND guarantees that the Z flag is clear. This way, both MARK and SPACE will time out by one-bit time.

The calling sequence, then, looks like:

```
BITZ    DEC COUNT
        BEQ EXIT
        LSR CHAR
        BCS DOMARK
        JSR SPACE
        JMP BITZ
DOMARK  JSR MARK
        JMP BITZ
```

What do we do when we have sent all eight bits and go to EXIT? We call JSR MARK one last time. That clocks out the enforced pause and sets the line back to idle for us. After this we can look for another character to send, or do other jobs if we want.

One odd protocol note: many programs choose to send the enforced pause – the stop bit – as the first part of a character. This is OK. So long as the pause is definitely sandwiched in between characters, it doesn't matter how you arrange it.

Receiving: An Outline

To receive, we have a slightly more complex task to do. When the line is idle, we must watch it constantly, since a character might begin at any time. Interrupts are sometimes used to good effect here.

When the start bit is spotted, we have a special job to do. At the moment that we detect the start, we're on the edge of the timing. If we delayed one-bit time, we might be at the beginning of the first data bit, or we might be a shade early and be at the end of the stop bit. This isn't satisfactory: we could read the wrong signal.

So instead of waiting one bit time, 3333 microseconds, we wait for one and one-half times: 5000 microseconds. That should place us right in the middle of the first bit timing, allowing us to take a solid reading. For the remaining bits, we'll return to a time delay of 3333, allowing us to check each bit smack in the middle of its time.

We'll pack the bits together by placing each bit into the Carry flag and then doing a ROR to the memory byte. And we'll remember to count, of course. When we receive the last bit, we will time out one more bit time; that should place us in the middle of the stop bit. We can now put our assembled character away and start watching the line for a new character.

It's educational and economical to do your own communications interface. We've looked at only one facet of the job: changing the computer's parallel data to a serial signal.

Of course, you could buy a UART (Universal Asynchronous Receive Transmit) chip to do the whole job for you, but that's the easy way out. Or is it? Last time I put a board together, my sports jacket became a smoking jacket. ©

It would be nice if you could just touch one key and then a BASIC program would immediately begin execution. Or if, when debugging a program, you could press the first function key and get a LISTing. Here's how to do it.

Although the program that allows the function keys to be programmable is in machine language, no knowledge of machine language is needed to use it.

Programming VIC's Function Keys

Jim Wilcox
Vienna, WV

Once this program is typed in, double check the DATA statements, since one error can result in a catastrophe. RUN the program after SAVEing it, and wait for about five seconds. The following should then appear: "F1 = ? ". Type in the BASIC command or statement you would like the function one key to equal. For every carriage return you would like, type in the back arrow located on the upper left-hand corner of the VIC. Once you are sure the function key has been defined properly, press the RETURN key. The program will then ask for the rest of the function keys' definitions. After you have defined the eighth function key, the computer will print READY. The function keys are now ready to be used. Just press the appropriate function key, and the characters for which it was programmed will be printed.

What If It Doesn't Work?

If the VIC just locks up or if you don't get the READY message, turn the VIC off and reLOAD the program. Recheck the program with the listing provided, from the beginning to line 65, especially the DATA statements.

When the READY message occurs after all eight keys have been defined and the VIC doesn't print the characters corresponding to the function key, check the program from lines 70 to 95.

If it still doesn't work, check the subroutine in lines 100 through 115.

How The Program Works

The BASIC program will POKE two machine language programs into your VIC. One goes into the cassette buffer, the other in the uppermost memory position. The program in the cassette buffer asks for the definition of each function key. Once the RETURN key is pressed, the program will store the ASCII value of the characters pressed in the uppermost portion of memory. After all eight keys have been programmed, the program will tell the computer to go to the other program in the top of memory every sixtieth of a second. The original program is not needed once the above operations have been performed and will be erased after any command for the cassette recorder is given. This is done to save 147 bytes of VIC's memory.

The second program will constantly check for a function key pressed. If one is pressed, the program will print the characters for which the function key was defined.

How To Save Memory

The longer each command for a function key, the more memory will be used up. If the commands are short, only about 200 bytes will be used up. The maximum amount of memory that can be used by this routine is about 800 bytes. To use the least amount of bytes, the commands can be typed in the shorthand method shown on pages 133-134 in the VIC *Owners Manual*.

Having programmable keys can be a great aid to a computer operator. The VIC is equipped with eight keys which you can use for whatever purpose you want. Time can be saved in writing and debugging programs. Here's an example:

```
RUN
F1=? LIST ←
F2=? POKE
F3=? RUN ←
F4=? PEEK(
F5=? GOTO
F6=? GOSUB
F7=? PRINT PEEK(7680) ←
F8=? LOAD ← LIST ←
```

```
5  F=0:C=PEEK(55)-120:IFC<0THENC=C
     +256:F=-1
10 D=PEEK(56)+F:POKE55,C:POKE56,D:
     CLR
15 S=828:I=146:GOSUB100
20 DATA32,198,3,165,55,133,251,133
     ,253,165,56,133,252,133,25
     4,169,49,133,0,169
25 DATA133,133,1,169,13,32,210,255
     ,169,70,32,210,255,165,0,3
     2,210,255,169,61
30 DATA32,210,255,169,63,32,210,25
     5,169,32,32,210,255,32,207
     ,255,72,160,0,165
35 DATA1,145,55,104,32,198,3,201,1
     3,240,14,201,95,208,2,169,
     13,145,55,32
40 DATA207,255,76,124,3,230,0,165,
```

NÜFEKOP

PROUDLY
PRESENTS
FOR THE VIC-20

3-D MAN

$12.95

THE EXACT MAZE FROM PROBABLY THE MOST POPULAR ARCADE GAME EVER WITH PERSPECTIVE ALTERED FROM OVERHEAD TO EYE LEVEL. THE DOTS. THE MONSTERS. THE POWER DOTS. THE SIDE EXITS. NEW ON SCREEN RADAR. THE GAME IS AMAZING.

OUR NEW 3D MAN REQUIRES, BUT WILL RUN WITH ANY, MEMORY EXPANDER.

EXTERMINATOR

By Ken Grant $16.95

FIRST THE BAD NEWS . . . THIS GAME IS LITERALLY FULL OF BUGS. THE GOOD NEWS? WE GUARANTEE HOURS OF EXCITING ENTERTAINMENT TRYING TO REMOVE THEM. SOME BUGS YOU ARE LIKELY TO COME UP AGAINST ARE SPIDERS, SNAILS, FLEAS, AND CENTIPEDES IN THIS RAPID FIRE, 100% MACHINE LANGUAGE, EXCEPTIONAL QUALITY GAME.

EXTERMINATOR RUNS IN STANDARD 5K VIC!

DEFENDER ON TRI

$12.95

AS PILOT OF THE EXPERIMENTAL DEFENDER STYLE SHIP "SKYES LIMITED" YOU ARE THE ONLY HOPE FOR ADVANCE PARTY OF SCIENTISTS TRAPPED IN ANCIENT ALIEN SPHERE WHICH SUDDENLY (HEAT FROM COLLISION COURSE WITH SUN PRESUMABLY - G.E.) CAME TO LIFE. 4 SCREENS WORTH OF UNIQUE DEFENSES, ON-OFF SHIELDS, FUEL DEPOSITS, ALIEN TREASURES, RUNNING TIMER, ENERGY, SCORE, AND VERY NICE GRAPHICS DISPLAY MAKE THIS ONE THAT DOES NOT QUICKLY WAX OLD.

DEFENDER ON TRI REQUIRES, BUT WILL RUN WITH ANY, MEMORY EXPANDER.

ANTIMATTER SPLATTER!

$12.95

A MORE DASTARDLY ALIEN COULD SCARCE BE FOUND THAN ONE WHO WOULD WIPE OUT AN ENTIRE CIVILIZATION BY DROPPING ANTI-MATTER ANTI-CANNISTERS, RIGHT? IF YOUR OPINION OF THIS ALIEN TROUBLE MAKER IS THE SAME AS OURS PROBABLY YOUR FIRST THOUGHT WAS, GET SOME MATTER! WE SAY CALM DOWN! ALL IS NOT LOST. A MOBILE RAPID SPLATTER CANNON CAPABLE OF BOTH BREAKING THROUGH HIS STANDARD ALIEN MOVING FORCE FIELDS AND LAYING WASTE TO THE EVER INCREASING NUMBER OF ANTI-CANNISTERS IS EVEN NOW HOVERING ABOVE US. IF ONLY OUR CANNONEER HADN'T CALLED IN SICK . . . SAY, WHAT ARE YOU DOING TODAY?

ANTI-MATTER SPLATTER IS 100% MACHINE LANGUAGE AND RUNS IN STANDARD 5K VIC.

OUR NEW CATALOG FREE! WRITE **NÜFEKOP** OR CALL 503-878-2113

GAMES WILL BE ON TAPE
UNLESS YOU REQUEST DISK

P.O. BOX 156 • SHADY COVE, OREGON 97539

MASTERCARD – VISA – C.O.D.

```
     Ø,41,1,2Ø8,1Ø,24,165,1,1Ø5
     ,4,133,1
45 DATA76,17Ø,3,56,165,1,233,3,133
     ,1,165,Ø,2Ø1,57,144,163,12
     Ø,169,LØ,141
5Ø DATA2Ø,3,169,HØ,141,21,3,88,169
     ,Ø,133,Ø,32,68,198,76,116,
     196,166,55
55 DATA2Ø8,2,198,56,198,55,96
6Ø S=PEEK(55)+256*PEEK(56):I=119:G
     OSUB1ØØ
65 SYS(828)
7Ø DATA165,Ø,24Ø,59,16Ø,Ø,177,251,
     32,L99,HØ,176,12,165,55,19
     7,251,2Ø8,21,165
75 DATA56,197,252,2Ø8,15,169,Ø,133
     ,Ø,165,253,133,251,165,254
     ,133,252,76,191,234
8Ø DATA166,198,177,251,157,119,2,2
     3Ø,198,32,L111,HØ,165,198,
     2Ø1,11,144,2Ø4,23Ø,Ø
85 DATA76,191,234,165,215,32,L99,H
     Ø,176,3,76,191,234,165,8,4
     1,1,2Ø8,247,16Ø
9Ø DATAØ,177,251,197,215,2Ø8,6,32,
     L111,HØ,76,L6,HØ,32,L111,H
     Ø,76,L81,HØ,2Ø1
95 DATA133,144,6,2Ø1,141,176,2,56,
     96,24,96,166,251,2Ø8,2,198
     ,252,198,251,96
1ØØ F=Ø:FORD=STOS+I:READA$:IFASC(A$
     )<58THENA=VAL(A$):GOTO115
1Ø5 IFASC(A$)=76THENA=VAL(RIGHT$(A$
     ,LEN(A$)-1))+PEEK(55):IFA>
     255THENA=A-256:F=1
11Ø IFASC(A$)=72THENA=VAL(RIGHT$(A$
     ,LEN(A$)-1))+PEEK(56)+F:F=Ø
115 POKED,A:NEXT:RETURN
```

©