NO. 58 MARCH 1983

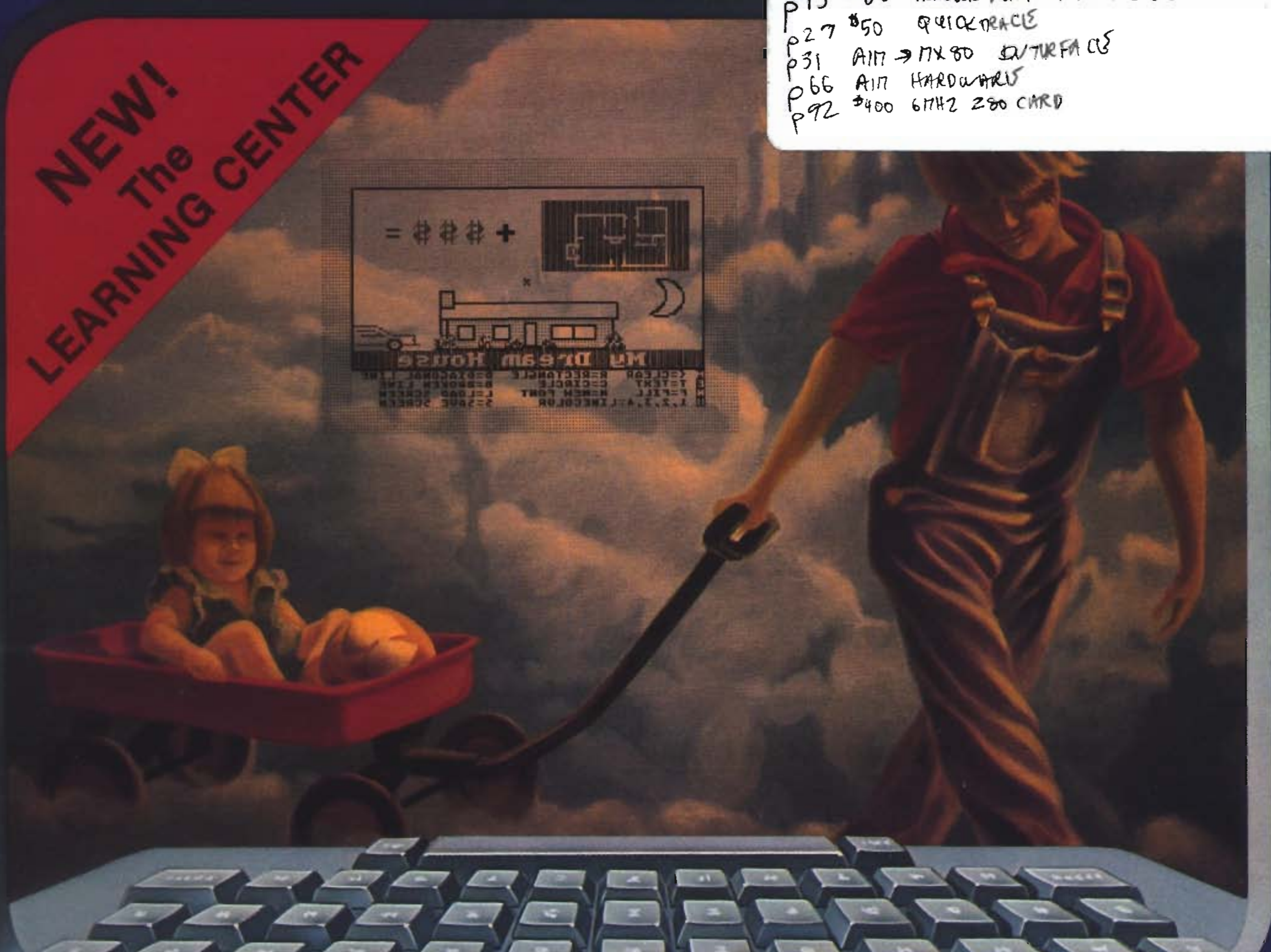
U.S./Canada Edition: $2.50
International Edition: $2.95
United Kingdom Edition: £2.00

MICRO™

Advancing Computer Knowledge

p13 $80 PRINTER PORT FOR VIC-20
p27 $50 QUICKTRACE
p31 AIM → MX 80 INTERFACE
p66 AIM HARDWARE
p92 $400 6MHZ Z80 CARD

NEW! The LEARNING CENTER



Printer Feature

Plotting with the VIC

APPLE PRINT-USING Routine

★ MICRO Calc for COMMODORE and APPLE ★

THE LEADING EDGE IN PRINTERS

ONE GREAT LINE. ONE GREAT WARRANTY.

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

THE PRO'S.

The Prowriters: business printers—and more. The "more" is a dot-matrix process with more dots. It gives you denser, correspondence quality copy (as opposed to business quality copy, which looks like a bad job of spray-painting).
Prowriter: 120 cps. 80 columns dot matrix compressable to 136. 10" carriage. Parallel or serial interface.
Prowriter 2: Same as Prowriter, except 15" carriage allows full 136 columns in normal print mode. Parallel or serial interface.

See Us At Booth #1146
COMDEX/SPRING '83
April 26-29, 1983
Georgia World Congress Center and
The Atlanta Apparel Mart
Atlanta, Georgia

THE STAR.

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

THE MASTER.

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

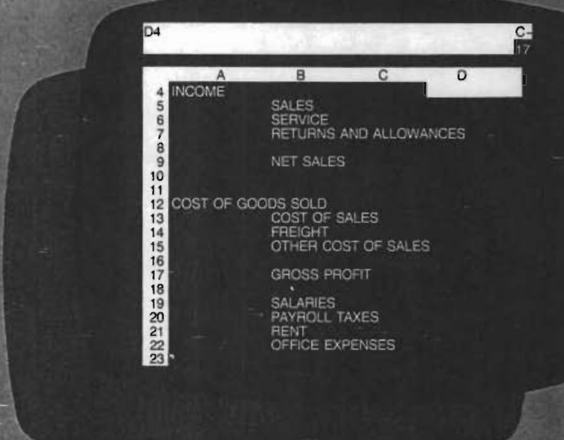
Distributed Exclusively by Leading Edge Products, Inc., 225 Turnpike Street, Canton, Massachusetts 02021. Call: toll-free 1-800-343-6833; or in Massachusetts call collect (617) 828-8150. Telex 951-624.

Circle No. 70

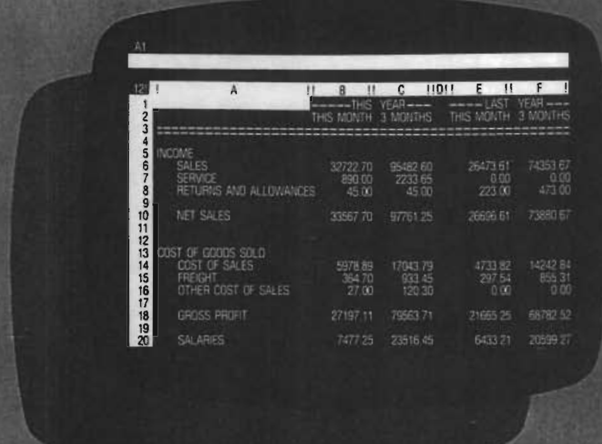
MAGICALC™

for the //e

The Visual Difference



VisiCalc®



MAGICALC™

See for yourself.

MAGICALC™—VisiCalc® COMPARISON

- MAGICALC does everything VisiCalc does
- MAGICALC does a lot more than VisiCalc does
- MAGICALC does most things faster than VisiCalc does
- MAGICALC does many things a lot easier than VisiCalc does
- MAGICALC loads VisiCalc files
- MAGICALC cost a whole lot less

MAGICALC FEATURES

MAGICALC has:

- Full compatibility with Apple //e
- 40-column standard video display
- 70-column upper- and lower-case video display
- 80-column board video display
- Automatic support of add-on memory boards (up to 512K)
- Individual column widths
- Protected fields and cells
- Invisible column
- Automatic page formatting (worksheets are automatically placed on paper)
- Fully compatible with VisiCalc and DIF files
- Command compatible with VisiCalc
- Command compatible with MAGIC WINDOW
- File loading by number

$149.95

artsci™
where art and science work for you

5547 satsuma avenue • north hollywood, california 91601 • 213/985-2922

VisiCalc is a registered trademark of VisiCorp. MAGICALC is a trademark of Artsci, Inc.

Circle No. 1

2MHZ 6809 SYSTEMS

GIMIX offers you a variety to choose from!

38 MB WINCHESTER SYSTEM $17,498.99

HARDWARE FEATURES:

- ★ 2MHz 6809 CPU
- ★ 512KB Static RAM
- ★ 8 RS232C Serial Ports
- ★ 2 Parallel Ports
- ★ DMA Double Density Floppy Disk Controller
- ★ Dual 8" DSDD Floppy Disk System
- ★ Dual Winchester Subsystem with Two19 MB 5¼" Winchester Drives

SOFTWARE FEATURES:

- ★ OS-9 LEVEL TWO Multi-User Operating System
- ★ OS-9 Debugger
- ★ OS-9 Text Editor
- ★ OS-9 Assembler

19 MB WINCHESTER SYSTEM $8998.09

HARDWARE FEATURES:

- ★ 128K Static Ram
- ★ 2MHz 6809 CPU
- ★ 19 MB 5¼" Winchester DMA Subsystem
- ★ 4 RS232C Serial Ports
- ★ 1 MB 5¼" Floppy Disk Drive
- ★ DMA Double Density Floppy Disk Controller

SOFTWARE FEATURES:

- ★ OS-9 LEVEL TWO Multi-User Operating System
- ★ OS-9 Text Editor
- ★ OS-9 Debugger
- ★ OS-9 Assembler

128KB MULTI-USER SYSTEM $6997.39

HARDWARE FEATURES:

- ★ 2MHz 6809 CPU
- ★ DMA Double Density Floppy Disk Controller
- ★ 128KB Static Ram
- ★ 2 RS232C Serial Ports
- ★ Dual 8" DSDD Floppy Disk System

SOFTWARE FEATURES: Your choice of either UniFLEX or OS-9 LEVEL TWO. Both are Unix-like Multi-User/Multi-Tasking Operating Systems.

56KB FLEX/OS-9 "SWITCHING" SYSTEM $4148.49

HARDWARE FEATURES:

- ★ 2MHz 6809 CPU
- ★ 56K Static Ram
- ★ 2 RS232C Serial Ports
- ★ DMA Double Density Floppy Disk Controller
- ★ 2 Built-in 5¼" 40tr DSDD Disk Drives (80 Track DSDD Drive Option . . add $400.00)

SOFTWARE FEATURES:

- ★ GMXBUG monitor — FLEX Disk Operating System
- ★ OS-9 LEVEL ONE Multi-tasking operating system for up to 56K of memory

WINCHESTER SUBSYSTEMS

Winchester packages are available for upgrading current **GIMIX** 6809 systems equipped with DMA controllers, at least one floppy disk drive, and running FLEX, OS-9 LEVEL ONE or OS-9 LEVEL TWO. The packages include one or two 19MB (unformatted) Winchester drives, DMA Hard Disk Interface, and the appropriate software drivers. The Interface can handle two 5¼" Winchester Drives, providing Automatic Data Error Detection and Correction: up to 22 bit burst error detection and 11 bit burst error correction.

Dual drives can be used together to provide over 30 MBytes of on line storage -- or use one for back-up of the other. (More convenient and reliable than tape backup systems.

#90 includes one 19MB Drive, Interface, and Software **$4288.90**
#91 includes two 19MB Drives, Interface, and Software **$6688.91**

Contact GIMIX for systems customized to your needs or for more information.

50 HZ Export Versions Available

GIMIX Inc. reserves the right to change pricing and product specifications at any time without further notice.

GIMIX® and GHOST® are registered trademarks of GIMIX Inc.
FLEX and UniFLEX are trademarks of Technical Systems Consultants Inc.
OS-9 is a trademark of Microware Inc.

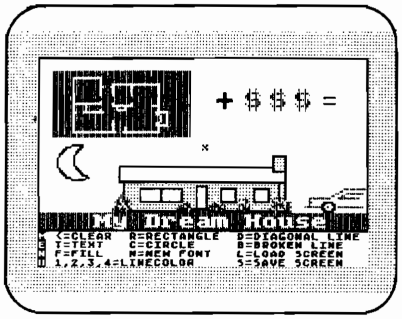
1337 WEST 37th PLACE
CHICAGO, ILLINOIS 60609
(312) 927-5510
TWX 910-221-4055

GIMIX Inc.

1982 GIMIX Inc.

Circle No. 2

About the Cover



Childhood Dreams is one of a series of original oil paintings by Frank Wyman for children's books. The computer industry is touching all of us — even children. MICRO's new section, "The Learning Center," caters to beginning computerists — including children. The graphic of a modern "castle" was generated by one of the programs appearing in the Learning Center ("Digi-Draft" by Tim Kilby, page 57).

Photo and painting by:
Frank Wyman
Wyman Art Studio
Lowell, MA 01852
617-459-7819

MICRO is published monthly by:
MICRO INK, Chelmsford, MA 01824
Second Class postage paid at:
Chelmsford, MA 01824 and additional mailing offices
USPS Publication Number: 483470
ISSN: 0271-9002

Send subscriptions, change of address, USPS Form 3579, requests for back issues and all other fulfillment questions to
MICRO INK
34 Chelmsford Street
P.O. Box 6502
Chelmsford, MA 01824
or call
617/256-5515
Telex: 955329 TLX SRVC
800-227-1617

Subscription Rates	*Per Year*
U.S.	$24.00
	2 yr. / $42.00
Foreign surface mail	$27.00
Air mail:	
Europe	$42.00
Mexico, Central America, Middle East, North Africa, Central Africa	$48.00
South America, South Africa, Far East, Australasia, New Zealand	$72.00

MICRO™

March Highlights

Printer Feature

Adding a printer to your computer system — or enhancing the capabilities of one you already own — will offer you versatility, convenience, and power. Many of the programs presented here expand the graphics, formatting, or control capabilities of a variety of printers.

You can make automatic pagination and user-selectable margins a part of the printer-driver routine; read John Vokey's "PRINT Control for Apple Printers" (p.24). Mr. Vokey presents a short machine-language subroutine that provides formatted output to most output devices. Larry Hollibaugh's MX-Driver assembly-language program ("Centronics Printer Driver for Your Microcomputer" p.31) will expand the use of your Centronics-compatible printer.

In "A PRINT-USING Routine for the Apple" (p.39), Celestino Monclova presents a machine-language routine that, when called from BASIC, merges two string variables (a mask and a data) with the following options: fixed decimal points and commas, fixed dollar sign and text, asterisk or dollar sign, and floating dollar sign.

If you are using an Apple printer interface card, you can send only 7-bit ASCII code. To find out how to add program control to the eighth bit, read Mark Boyd's "A Full Byte for Your Apple Printer" (p.42).

We also offer a brief tutorial by Fred Wallace, "Plotting with the VIC" (p.19), that describes the graphics mode available on a VIC-20 equipped with the VIC-1515 graphic printer. Mr. Wallace includes a BASIC subroutine that permits plotting of either a mathematical function or a user-generated array on a grid with labeled axes.

Robert Paul explains how to utilize the open area of OS-65D directory sector 1 to include a diskette ID on printed directories in "Disk ID for Printed Directories" (p.36).

If you're looking for a printer, read through our Information Sheet (p.105) before you go shopping. It describes the 10 to 20 most popular printers for the computers we cover; major characteristics are compared.

New! The Learning Center

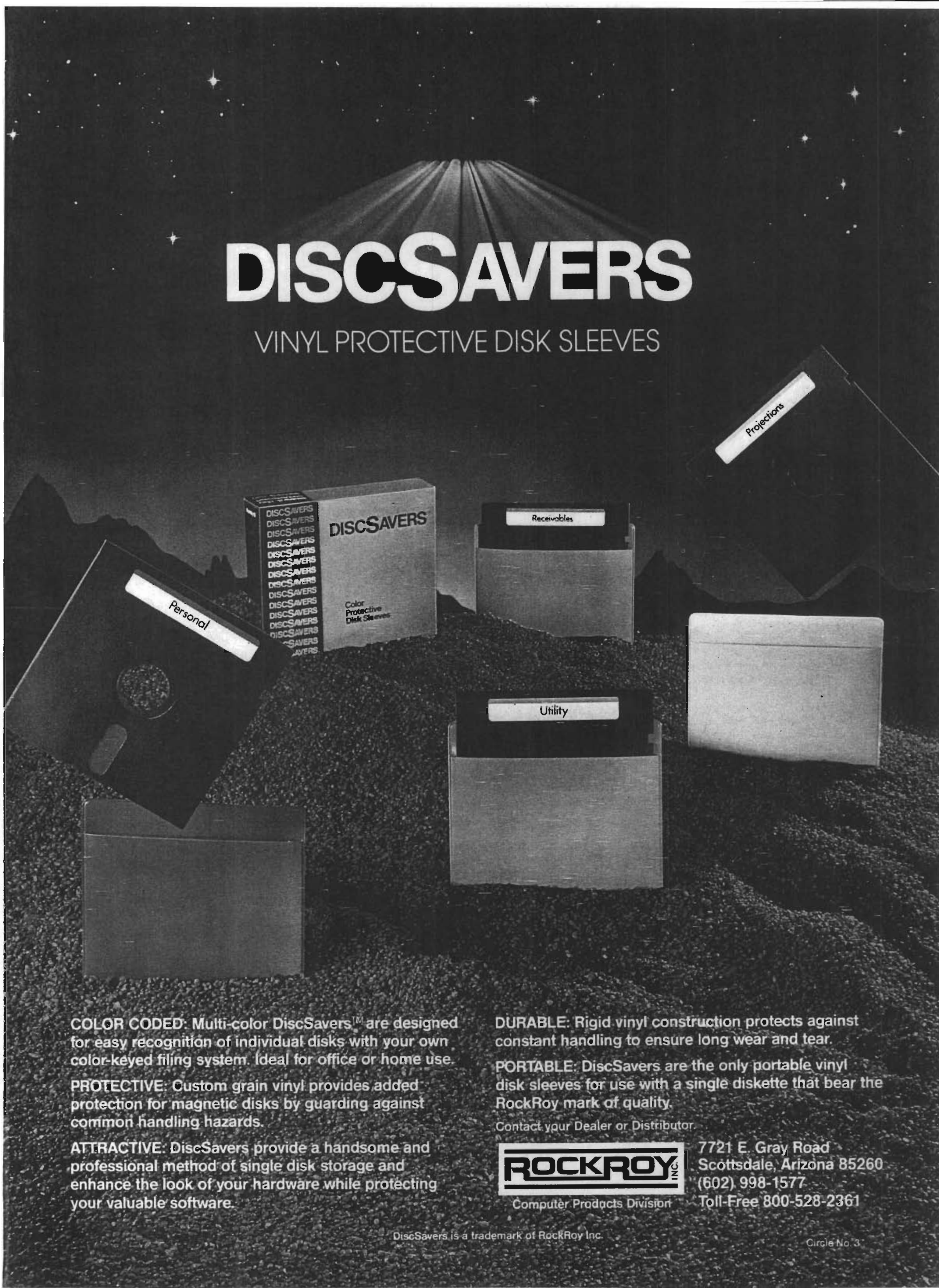
Now computerists just beginning to tap the keyboard can enjoy MICRO, as well as sophisticated programmers. *The Learning Center* provides fun, easy-to-key-in programs, with accompanying tutorial text, that teach you programming techniques and concepts. Among the articles appearing in *The Learning Center* this month are "MICRO Calc for Commodore and Apple" by Loren Wright (p.47). This miniature spread-sheet program lets you perform a variety of calculations. Minor changes are provided for all Commodore machines; Phil Daley offers an equivalent program for the Apple.

"Digi-Draft" by Tim Kilby (p.57), a drawing program, allows you to construct images on an Atari GRAPHICS 7 screen. Learn commands to change colors, load and save, draw points, lines, circles, and rectangles.

Another entertaining program displays messages in large letters across your screen. "BANNER: A Display Program for the TRS-80C" by Bryan Christiansen (p.65) also includes powerful machine-language loading techniques that provide efficient handling of machine-language subroutines.

As Always...

Don't miss our usual machine-specific coverage in Apple Slices, CoCo Bits, From Here To Atari, and Pet Vet.


DISCSAVERS
VINYL PROTECTIVE DISK SLEEVES
DISCSAVERS
Color Protective Disk Sleeves
Receivables
Projections
Personal
Utility
COLOR CODED: Multi-color DiscSavers™ are designed for easy recognition of individual disks with your own color-keyed filing system. Ideal for office or home use.
PROTECTIVE: Custom grain vinyl provides added protection for magnetic disks by guarding against common handling hazards.
ATTRACTIVE: DiscSavers provide a handsome and professional method of single disk storage and enhance the look of your hardware while protecting your valuable software.
DURABLE: Rigid vinyl construction protects against constant handling to ensure long wear and tear.
PORTABLE: DiscSavers are the only portable vinyl disk sleeves for use with a single diskette that bear the RockRoy mark of quality.
Contact your Dealer or Distributor.
ROCKROY INC.
Computer Products Division
7721 E. Gray Road
Scottsdale, Arizona 85260
(602) 998-1577
Toll-Free 800-528-2361
DiscSavers is a trademark of RockRoy Inc.
Circle No. 3

MICRO™

Advancing Computer Knowledge

DEPARTMENTS

2 March Highlights
7 Editorial
8 CoCo Bits
10 From Here to ATARI
12 Short Subjects
14 Apple Slices
90 PET Vet
92 Interface Clinic
94 Reviews in Brief
97 Software Catalog
100 Hardware Catalog
101 Language Information Sheet
103 Data Sheet
105 Printer Information Sheet
111 6809 Bibliography
112 Next Month in MICRO
112 Advertiser's Index

PRINTER FEATURE

19 Hi-Res Plotting with the VIC .... *Fred Wallace*
Combine VIC-1515/1525 printer with the VIC-20 for high-resolution graphics

24 Print Control for APPLE Printers ..... *John R. Vokey and H. Cem Kaner*
A short subroutine for automatic pagination and user-selectable margins

31 Centronics Printer Driver for Your Microcomputer .... *Larry R. Hollibaugh*
An MX-Driver assembly-language program

36 Disk ID for Printed OSI Directories .... *Robert A. Paul*
Put unused disk space to good use

39 APPLE PRINT-USING Routine .... *Celestino R. Monclova*
A machine-language routine for formatted output

42 A Full Byte for Your APPLE Printer .... *Mark J. Boyd*
Greater control using the eighth bit

THE LEARNING CENTER

47 MICRO Calc for Commodore and APPLE .... *Loren Wright*
A miniature electronic spreadsheet

57 Digi-Draft for Atari 400/800 .... *Tim Kilby*
A sophisticated drawing program

61 The Computer Revolution Reaches the Community .... *Emmalyn H. Bentley*
A look at a new microcomputing learning center

65 BANNER: A Display Program for the Color Computer .... *Bryan Christiansen*
Display a variety of messages in large type on your screen

70 Glossary of Computer Terms

ARTICLES

73 68000 BCD and Privileged Instructions .... *Joe Hootman*
Our 68000 series continues

77 A Versatile Hi-Res Graphics Routine for the APPLE. *Adam P. King*
Design multicolored shapes

82 APPLE Disk Track Copy for Non-Matching Numbers .... *Roland E. Guilbault*
Increase your disk library's efficiency

85 It's All Relative, Part 4 Using Commodore's Relative Records .... *Jim Strasma*
Read and write relative disk files

Circle No. 4

for the Artist

Special Effects

by David Lubar and Mark Pelczarski

Paint on your computer in over 100 colors with 96 different brushes! Magnify mode for precision touch-ups; mirror image; color reverses; move any part of picture anywhere; packing routine for increased disk storage of any standard graphics screen. Can be used by anyone. Paddle/joystick $39.95. Apple Graphics Tablet $69.95.

for the Designer

THE COMPLETE GRAPHICS SYSTEM II

by Mark Pelczarski

Everything needed for computer-aided design. Easily draw and design in two or three dimensions, mix text and graphics freely, define shapes and create typefaces with unique character generator. Great for presentations, videotape displays, storing designs on disks, graphics for other programs, or just for experimentation. Paddle/joystick $69.95, Apple Graphics Tablet $119.95. 50 additional fonts, $19.95.

for the Programmer

The Graphics Magician

by Mark Pelczarski, David Lubar, and Chris Jochumson

Add fast, smooth animation and hundreds of pictures to your programs. Create animated shapes and draw paths; have up to 32 independently controlled figures on screen. Use a special palette of over 100 colors to create pictures and objects in highly compressed format. Easy to use, and includes a special programming tutorial. Has saved months in development time for several games already on the market. Paddle/joystick $59.95, Apple Graphics Tablet $69.95.

penguin software 830 4th Avenue, Geneva, Illinois 60134 (312) 232-1984

All Penguin applications products are now on unprotected disks for your convenience. Apple II is a trademark of Apple Computer, Inc.

MICRO™

Editorial

"The more things change, the more they...uh...change!"

Everyone knows that the way that old French proverb is supposed to end is, "...the more they stay the same."

But the fact is that's never been true at MICRO magazine. Change is change. It's accepted because it's necessary. The microcomputer industry itself is far too dynamic to tolerate anything static for very long.

And, of course, MICRO has been around for a while; almost as long as the industry itself (MICRO is now in its sixth publishing year), which means it's seen its share of changes.

In that regard, however, our March 1983 issue is a benchmark; you'd have to go back a long, long way before you found an issue of MICRO that contained so many changes (and the promise of even more to come).

Start with this month's cover. Maybe you noticed the image inside the screen is a painting, not a photograph. A small change, but an important one. The work is by Frank Wyman, an exciting artist who's agreed to lend his talents to MICRO on a regular basis.

The cover art serves to introduce a brand new section in MICRO, "The Learning Center," but don't be mislead by the youngsters it depicts. This new section is definitely not "kid's stuff." It's MICRO material presented just a little bit differently — more clearly spelled out, with some information gaps filled in.

The new section is more tutorial and, generally speaking, plans are to keep its focus on programming information for the Atari 400/800, Commodore 64 and VIC-20, TRS-80C, and Apple. But you may find similar information on other systems like the TI-99/4A and Timex/Sinclair from time to time.

We see this section as a way of breaking new editorial ground; we're taking MICRO's approach to computers and programming and applying it to these relatively new systems.

Think about that for a second. The possibilities are truly intriguing. The "Visicalc-type" program in this premier edition of "The Learning Center" is a good example.

Another alteration you might notice in this issue is the new page of Reader Service Cards we've bound into it. The top two are self-explanatory. The card on the bottom is designed to be used with the individual advertisements you'll find in this issue. There's a number on or near each ad. If you want to receive more information about a product you see advertised in this issue, just circle the right number, fill in your mailing information, and drop the card in the mail.

Throughout this issue and in issues to come, you'll find MICRO using more color than in the past. We believe that color adds an important dimension to the magazine. For example, in a wiring diagram color can clarify and deliver the information far more efficiently than the written word. It is our hope to use color extensively in future issues of MICRO — for diagrams, colored screen shots of running programs, and even for its pure cosmetic effect in feature articles.

So, what we have here in the March 1983 issue of MICRO magazine is the beginning of a new look, a new section, and a new service. A fair amount of innovation, I'd say, for a single issue.

April, of course, is another story....

Oh yes, one more thing. I'd like to introduce myself — MICRO's new publisher.

John Grow

CoCo Bits

By John Steiner

In my January column (MICRO 56:92) I mentioned the TDP 100, a Color Computer built by Radio Shack for distribution through other dealers. This month's column was written using my new TDP 100 with disk Scripsit. The System 100 computer is presently available in stock configuration as a 16K computer with either standard or Extended BASIC. In addition, it is available from some locations as a 32/64K model. Retail prices are comparable to Radio Shack prices. My machine is equipped with the 64K RAM mode modification provided by Computerware, Inc. As I become more adept at using the 64K capacity, I will pass the information along.

Technically, the computer is software-compatible with the TRS-80 Color Computer. The circuit board, however, is altogether different in layout; the familiar components are there but in different locations. This may cause problems with hardware accessories built for insertion into the computer.

On the "E" board model TRS-80, memory jumpers are labeled 16K and 32K. On the 100 model, they are marked 16K and 64K. Does this indicate a possible marketing change? Making a 64K machine available would cost little, yet provide an extra sale point.

The TDP 100 comes with a white case, built slightly higher than the TRS-80. It has long cooling slots in the back of the cabinet into which a loose wire end or coin might easily slide. When my warranty expires, I will probably glue some nylon netting to the underside of the cabinet to forestall any accidents.

I have noticed that the colors displayed on the screen are slightly different from my other machine. Adjusting the hue control on the TV does not restore the colors to those with which I am familiar. I don't know if there is a difference in the color circuitry, or if my sample happens to be adjusted differently. Despite these minor problems, my impressions of the computer have been favorable so far.

While on the subject of CoCo "clones," there are other computers available or in the works. In England, the Dragon is similar to CoCo, although it is not completely compatible with either hardware or software. A Japanese company is making a compatible computer that probably will be available by the time you read this. There are rumors of still another compatible machine, but I have no information on that. There should be no shortage of software for CoCo.

I received a letter from Maury Mead asking about the Supercharger board available from Spectral Associates. This board accesses map type 1, the all RAM mode, without modifying the computer. He would like to know if the board will work with Frank Hogg's FLEX. If you have any information on this, please let me know.

Ken Christiansen of Fargo, North Dakota, provides the following information regarding tape ASCII files. If you have an I/O error in a tape ASCII file, or if you have accidentally recorded over the header, there is a way to recover at least some of the data. You can experiment by loading a program and resaving it using the ,A option. CSAVE "filename",A saves the program in ASCII format rather than tokenizing the keywords. As the tape is being written, you will notice that it contains blocks of data. Watch the record indicator; you will see the spindles actually stop turning between the blocks. That is the key to recovering the data.

If you accidentally record over a file header, trying to load the information only results in an I/O error message. Put in a tape that contains a valid file (its filename is unimportant) and begin the loading process, watching the record indicator and tape spindles. When the recorder stops, immediately pull out the earphone connector cable. The computer will not hear the rest of the incoming data. Now you can stop the recorder and reload your damaged file. Disconnect the motor remote plug and position the tape to the beginning of the file. Press PLAY and listen as the file is being read. There will be a slight gap between data blocks. When you first note the silence in the gap, quickly reinsert the earphone plug. When the next block starts, the computer will accept it as a continuation of the previous file. At this point you can use a text editor, or possibly your file program itself, to edit out the undesired information from the first file block. This method also erases any data contained in the first block.

You can restore program files saved in ASCII format that have an I/O error in the center of the file. CLOAD the file and count the number of blocks until the I/O message appears. Next, use CLOAD to bring in the file again, counting the incoming blocks. When you reach the block number just before the damaged block, pull out the earphone plug. Using the technique described earlier, skip over the damaged block. When that block is finished, quickly plug in the earphone cable. The program will continue loading. You will have to reconstruct only the data in the damaged block. Though I haven't done much experimenting with the technique, I was able to merge two programs together.

I have some more memory locations and functions for you this month. The start address of a BASIC program is located in memory locations 25 and 26. The end address is stored in locations 27 and 28. An interface with a machine-language routine could be done easily with this information. You can CSAVEM the BASIC program, along with a machine-language driver. The driver could auto-execute and do a JMP to $AE75, the start address of the RUN routine. This procedure creates an auto-executing BASIC program.

You may contact John at 508 Fourth Ave., N.W., Riverside, ND 58078.

MICRO™

MICRO™

From Here To Atari

By Paul S. Swanson

The recently announced Atari 1200 computer should be available soon at a suggested retail price of $899.00. The significant differences between it and the 400 and 800 computers include 64K memory, more function keys, and changes in the operating system.

The operating system of the Atari 1200 supports four more graphics modes. These modes are instruction register modes 4, 5, C, and E. IR modes 4 and 5 are character graphics modes, which allow characters to be formed using bit pairs as color register references. IR mode 4 uses one scan line per character row and mode 5 uses two. Most of the reference materials I have claim that these are four-color modes, and it is true that you can have only four colors in each character on the screen, but they are actually five-color modes. The colors normally originate from registers 4 (background), 0, 1, and 2, but if you PRINT the character in inverse video the color that would have been from register 2 will be taken instead from register 3.

IR modes C and E are both map modes. Mode C is a two-color mode using 160 dots per line; each mode line is a single scan line. Mode E is like OS mode 7 (which is IR mode D) except it has twice the vertical resolution, using one scan line per mode line instead of two. These new modes can be declared with GRAPHICS statements from BASIC as modes 12 through 15.

There are only 14 modes available in the Atari hardware (GTIA modes are actually all versions of mode 8). The Atari 1200 operating system supports all but the IR mode 3, a special character mode that supports lower-case descenders. It has ten scan lines per character, allowing you to use all eight in the character set because the other two are automatically blanked. IR mode 3 is the only mode that I have not found any reason to implement and can't recall seeing any programs that use it. If you want to implement IR mode 3 you need a complete custom character set. It is fairly simple to alter a mode 0 display list to use IR mode 3.

The Atari 1200 has only two joystick ports, using the locations of PORTA (PORTB doesn't exist). The new arrangement of the keyboard makes more sense. The BREAK key is no longer on the keyboard next to BACK S, which should eliminate unintentional BREAKs while you are trying to run your programs.

The new function keys, as well as most of the keys on the keyboard, are programmable. There are three 64-byte maps in memory that define the code for the key alone, SHIFT plus the key, and CTRL plus the key.

Like the Atari 400, the Atari 1200 has only one cartridge slot. Some third-party cartridges will not work on the 1200 because there is a slight difference in the physical dimensions of the slot, but the Atari cartridges will work fine.

Except for situations where the software calls for game controllers in ports 3 and 4 and for cartridges in the right slot, the 1200 should be fully upwards compatible as far as software. If you obeyed the restrictions outlined in the operating system manual instead of looking for ways to "cheat" in the operating system listing, your software should work on the 1200.

Restoring Registers

The vertical blank interrupt routine is in two parts. The first part updates system clocks and the second part shadows the hardware registers. The second part, referred to as the deferred vertical blank interrupt routine, is easily disabled — just POKE 66,1.

Some interesting effects can be produced by playing with this feature. For example, instead of watching a screen fill up as the program draws it, you can blank it out or put some other message on it, then POKE 66,0 and your completed screen instantly appears. Just follow these simple steps:

1. POKE 66,1 to disable the deferred vertical blank interrupt routine.
2. POKE 54272,0 to blank out the screen, or set up a display list and a screen somewhere in memory that is not used by the OS screen and POKE the display list pointers into 54274 (lo) and 54275 (hi). POKE color and other information directly into the hardware registers — not the shadow registers.
3. Use the standard BASIC statements (or the shadow registers in machine language) to set up the next screen. Do not use hardware registers directly.
4. POKE 66,0 and your new screen will immediately appear on the television.

Normally, you do not need to set the address for the Jump on Vertical Blank (JVB) instruction because the shadowing restarts the display list from the pointers in the shadow registers. If you disable the deferred VBI routine, you must supply the address or ANTIC will get lost in memory and not give you the interim display you set up.

If there are many computations and screen commands required to complete the final screen, you will probably want to supply the display list and screen so that there is an interim screen showing during the computations. Remember that DMA (Direct Memory Access) can steal up to 30% of your processing time. If you make the interim screen simple, you limit the DMA substantially. One or two lines of IR mode 6 or 7 (OS modes 1 and 2) instead of a full graphics or text screen will not steal many cycles from your computations.

There are other ways to shut off the vertical blanking. You can write directly to the hardware register NMIEN at 54286 (write a zero), but that stops the entire routine and the clocks will also stop. If you do this, write a 64 to 54286 to reenable the routine. Using location 66 (called CRITIC) does not stop the system clock.

What happens in the above procedure is that shadowing is temporarily suspended. This means that you can modify the normally shadowed hardware registers in any way you want to get any interim effect you need, then restore all of them to the values you set up for the next screen by a POKE 66,0. All of the hardware registers are updated during a vertical blank period, so the next television frame contains your new screen. You do need your own display list and screen area because your GRAPHICS, PLOT, DRAWTO, and PRINT statements will still change the memory area that the operating system "thinks" the screen occupies. SETCOLOR, as well as SOUND and a few other statements, write to shadow registers, so they simply set up the shadow registers for the new screen without affecting the current screen.

Compatibility Note

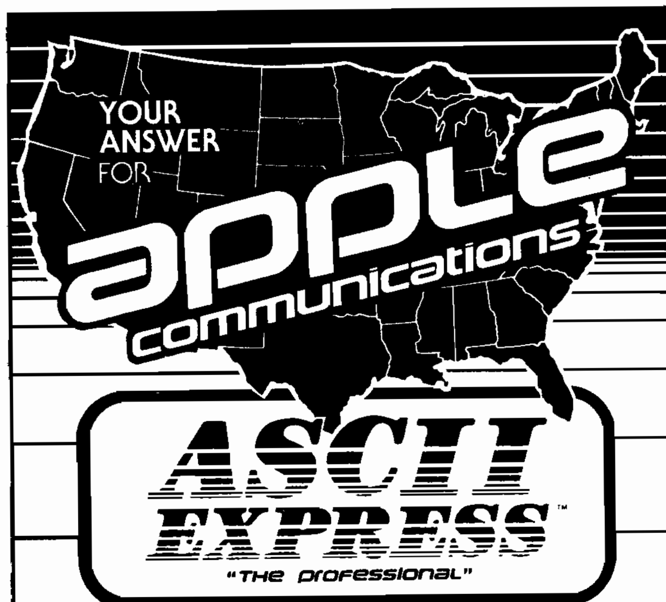
The Commodore 64 has several features that are similar to those found on the Atari. One external feature is the pair of control ports. The 64 has two jacks that are pin-compatibile with the Atari controller jacks, so any device set up for the Commodore 64 control ports will probably work without alteration on the Atari controller jacks. Commodore's PORTA and PORTB are not the same as the Atari PORTA and PORTB, so you will have to reshuffle the software a little. I recently had my Atari computer "talking" to a Commodore 64 through these controller ports and the differences were mainly the method of setting up the ports for mixed input and output and the memory location of the ports themselves.

Next Month

In the interest of supporting the new operating system modes of the 1200 without abandoning the 400 and 800, I will review setting up character sets for IR modes 4 and 5. The information will be compatible with the Atari 400 and 800. If you want to implement it on the 1200, you can either do it the same way or take advantage of the OS support, which will simplify your program in a few places.

If you have any topics that you would like discussed in this column, write me at 97 Jackson Street, Cambridge, MA 02140.

MICRO



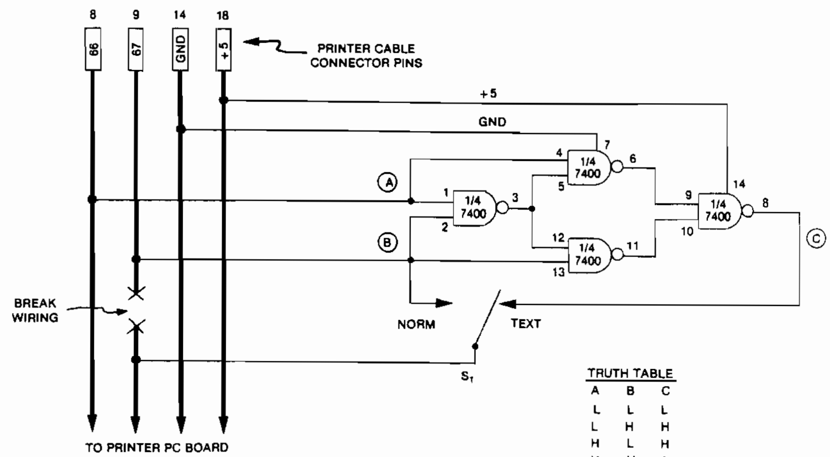
MICRO™ Short Subjects

$2 Lower-Case Fix for the Axiom 801P — *by Louis F. Sander*

Raising Numbers to a Power with Pascal — *by Robert D. Walker*

$2 Lower-Case Fix for the Axiom 801P

Louis F. Sander, 153 Mayer Dr., Pittsburgh, PA 15237



NOTES: 1. HEAVY LINES ARE ORIGINAL PRINTER WIRING.
2. EXCLUSIVE-OR GATE CIRCUIT IS FROM RADIO SHACK ENGINEERS NOTEBOOK, 1980 EDITION, PAGE 52.
3. IN TEXT MODE, WHENEVER BIT 6 IS HIGH, THE CIRCUIT INVERTS BIT 7, EXCHANGING CODES 64_{10}-127_{10} & 192_{10}-255_{10}

Figure 1

The Axiom 801P was one of the few printers with built-in PET graphics and became a popular companion to the original PET computer. Today, since little else in its price range can produce such excellent listings and graphics, the 801P is still widely used. But using it with late-model PETs creates a problem: with any but original ROMs, the printer reverses upper- and lower-case characters when it is in lower-case mode. Refer to figure 1 as you read my description of the $2.00 add-on circuit that corrects this problem.

With S1 in the NORM position, the printer's function is unmodified. But when S1 is flipped to TEXT, the blocks of characters with decimal codes 64-127 and 192-255 will be swapped (e.g., a CHR$(64) will print as a CHR$(192), a CHR$(192) as a CHR$(64), etc.). With the printer in graphics mode this creates a nightmare, but in lower-case mode, the upper- and lower-case alphabets are put where they belong. As an unwanted side effect, six keyboard-generated characters will now print incorrectly, but they are rarely used in actual text. (They are the square brackets, at sign, backslash, up and left arrows.)

Construction of the add-on circuit is simple and noncritical. All five connections to the printer can be made where the 36-pin printer cable connector mounts to the green circuit board. You can break wire XX by gently desoldering pin 9 from the board, and holding it out of the way with a drop of epoxy. To keep things small, I glued the 7400 chip to the side of the toggle switch and soldered wires directly to the pins, mounting the switch in a hole in the side of the printer.

Once the circuit is in place, turn S1 to NORM and run the short program listed here. Then flip to TEXT, run the program again, and compare the two printouts. (Be advised that CHR$ 96-127 and CHR$ 224-255 are not generated from the keyboard, so their swapping will have little practical effect.) In use, you will usually keep S1 in the NORM position, where the printer will work in standard fashion. When you want to print upper- and lower-case text, switch to TEXT and send a CHR$(14) to the printer to put it in lower-case mode. When you see your capital letters right again at last, you'll know your $2.00 was well spent, and that your little printer has a new lease on life.

```
 50 REM *** AXIOM PRINTER TEST ***
 60 REM
 70 REM PRINTS ALL PRINTING CHARACTERS
 80 REM IN GRAPHICS & LOWER-CASE MODES
 90 REM
 100 OPEN4,4:CMD4:PRINTCHR$(8)CHR$(12)
 110 G$=CHR$(15)+"      ":L$=CHR$(14)+"      "
 120 PRINT"CHR$   G L   CHR$  G L"
 130 FOR I=32TO127
 140 IFI<100THENPRINT"     ";
 150 PRINTI;G$CHR$(I)L$CHR$(I)"       "I+128
G$CHR$(I+128)L$CHR$(I+128)
 160 J=J+1:IFJ=16THENPRINT:J=0
 170 NEXT
 180 PRINT#4:CLOSE4
```

Raising Numbers to a Power with Pascal

Robert D. Walker, 2850 Delk Rd., Apt. 2B, Marietta, GA 30067.

My first introduction to Pascal was on the Apple computer about two years ago. Prior to this I had done most of my programming in BASIC. After becoming familiar with Pascal, however, I have rarely used any other language. Among other things, Pascal makes it easy to logically structure programs and keep track of variables. One shortcoming, however, had me wondering whether this language was suitable for scientific purposes — there was no direct way to evaluate a number raised a power. But after remembering some simple laws of logarithms, I found the answer.

Pascal contains the function 'EXP(Z)', which raises the base (approximately 2.718) of the natural system of logarithms to the Z power. Using the following logic, you can develop a formula that uses the EXP(Z)

Short Subjects *(continued)*

function to evaluate Y raised to the X power. The initial expression is:

$y^x = e^z$

Using laws of logarithms yields

$x \ln y = z$

Substituting for z in the initial equation yields

$y^x = e^{x \ln y} = EXP(X*LN(Y))$

This formula can be used in a Pascal function to directly evaluate Y raised to the X power (see listing).

Although this solution is indirect and takes significantly more processing time than a similar machine-language subroutine, I find that for my purposes Pascal supports scientific programs admirably.

```
{ THIS FUNCTION RAISES Y TO THE X POWER }
{ Note: Apple Pascal users must USE(S)   }
{       the TRANSCEND library unit in    }
{       their main program.              }

FUNCTION EXP2(X,Y: REAL): REAL;
BEGIN
  EXP2: = EXP(X*LN(Y))
END;
```

MICRO

MICRO™

Apple Slices

By Tim Osborn

This month I present a file structure known as linked-lists. This structure chains records together into logical groups so that the records in a group (or set) can be accessed in a predetermined sequence. Items within the set may be sequenced by some key field, such as name, or by the order in which the items are entered into the set. The file may contain many sets of records linked together, or it may be composed of just one set. The records are chained together by including a pointer to the next and/or previous record in the set. The pointer, simply a record number, is carried as a field in each record of the set.

Many design options are available when you work with linked-lists. For example, records can belong to more than one set, and there may be multiple record types within each set. This month's programs (LINK-CREATE and LINK-PROG) use a simple file structure containing two sets. I chose to include the second set simply to manage the free space in the file. By chaining together unused records, you can eliminate the necessity of performing what is commonly called garbage collection (a reorganization of the file).

When the file is initialized (by LINK-CREATE) it contains two record types: the header record (there is only one) placed at record zero, and the data records. The header record contains two fields: FIC (first-in-data-chain) and NA (next-available-record). The data records contain the following fields:

1. FIRST$ — First name
2. LAST$ — Last name
3. STR$ — Street address
4. CITY$ — City address
5. ST$ — State address
6. ZIP$ — ZIP code
7. TEL$ — Telephone number
8. NXT — Next record in chain

NXT is the pointer to the next record in the set. When LINK-CREATE initializes the file it sets all the data fields to nulls (fields 1 through 7), and the NXT pointer points to the n + 1 record, where n is the current record number. Thus, LINK-CREATE builds a chain of available records. The first record is pointed to by NA in the header record (NA is initialized to 1). LINK-CREATE prompts the user for the file size desired and places that many data records in the file. The last record in the chain contains a zero in the NXT field. Zero signifies an end of chain; thus zero is also placed in FIC since the used record chain is empty when the file has just been initialized.

LINK-PROG initializes the I/O arrays in line 10. I use element 0 as the file input area (OLD is used as the subscript), element 1 is the keyboard or user input area (NW for "new" is used as the subscript), element 2 is the file output area (WR for write is used as the subscript).

Lines 100-145 ask for the desired function (A(DD), C(HANGE), D(ELETE), I(INQUIRE), (Q(UIT)) and then branches to it. Take a look at how these subroutines work:

Line	Description
5000	Reads the data records into the old elements.
6000	Gets user inputs from the keyboard and places them in the NW elements. For the add function (A$ = "A") it prompts the user for all data fields. For other functions it asks for only the first and last name.
7000	Writes the data records from the WR elements.
8000	Moves data fields from elements F (from) to elements T (to).
9500	Displays data fields from the old elements.

ADD (lines 1000 - 1520)

The add function first reads the header record by a GOSUB 4000. It then checks in line 1010 to see if NA is equal to zero. If there is more free space the user is prompted to enter their various data fields by a GOSUB 6000. Line 1020 reads the available record by setting R=NA and a GOSUB 5000. Line 1020 also saves the new record number for future use by NR=NA. Line 1030 resets NA to be equal to NXT of the new record. The record previously pointed to by the new record becomes the next available. If NXT is equal to zero then the new record was the last available record and NA is set to zero. Line 1040 checks to see if FIC = 0, which means this new record is the first and only record in the used record chain. Therefore, FIC is set to the new record's number (NR) and the next pointer of the new record is set to zero (NXT (NEW) = 0).

The file is searched by starting at FIC (line 1050) and checking each successive record to see if the new record's key is less than the key of the record to which it is being compared (line 1060).

If the new record's key is more than the key of the record to which it is being compared, then line 1070 checks to see if the current record is at the end of the data chain (NXT(OLD) = 0). If so, then the current record is rewritten, making its NXT pointer equal to the record number of the new record (NXT (OLD) = NR). The NXT pointer of the new record is set to zero as this record is now at the end of the data chain. A GOTO 1500 then rewrites the header record and creates the new record.

On the other hand, if the current record is not at the end of the data chain, then it is saved as the last record compared to in setting LR = R in 1075. The program prepares to access the next record in the chain by setting R = NXT(OLD) and performing a GOTO 1055 in line 1080.

If the new record's key is less than the key of the record being compared, then the proper position for the record has been found, and the program falls through to line 1061. Line 1061 sets the next pointer of the new record to the current record number. Line 1062 checks to see if the current record is equal to FIC, in which case FIC is reset to the new record number and the header record and new record are written by a GOTO 1500. If the current record number is not equal to FIC then it is necessary to update the previous record to point to the new record, which inserts the new record somewhere into the middle of the chain. Then the header and new record are written (line 1064).

CHANGE (lines 2000 - 2320)

The change function reads the header record (line 2004). Line 2005 checks to see if FIC = 0. If FIC is not equal to zero the user is asked if the name is to be changed. If the user responds "Y" (yes), the program proceeds to delete the record (GOSUB 3030 in line 2030). Since the delete routine saves the deleted data in the NW elements, line 2031 moves this data back to the OLD elements. Lines 2032 through 2038 accept the new first and last names from the user. Control is then passed to line 2070.

If the user does not want to change the name then the change function continues at line 2050 where a GOSUB 6000 gets the first and last name, and a GOSUB 9000 searches for the desired record.

Line 2070 checks to see that the record was found; if not (RC>0) then a 'not found' message is displayed. Lines 2080 through 2297 prompt the user to change any of the data fields in the record. Line 2300 checks for a name change (CN$ = "Y"). If there was, the data is moved from OLD to NW and the record is added through a GOTO 1020. If the name was not changed then the record is rewritten (line 2310) and control returns to the main menu.

DELETE (lines 3000 - 3130)

The delete function first reads the header record (line 3010), then makes sure FIC = 0. Line 3030 GOSUBs to 6000 to get the key of the record to delete. Line 3040 GOSUBs to 9000 to search for the record. If the record is found (RC = 0) then processing continues.

When the record is found, line 3045 saves the deleted record number in DR then saves the data in line 3045. Line 3060 makes sure the deleted record is the first in the chain (FIC = R). FIC is set equal to the next pointer of the deleted record. If the deleted record is not first in the chain, lines 3070 through 3080 update the previous record's NXT pointer to point at the record at which the deleted record originally pointed. Lines 3100 - 3120 make the next pointer of the deleted record equal to the next available record (NXT(WR) = NA), and the next available equal to the deleted record number (NA = DR). This pushes the deleted record onto the top of the available record stack.

INQUIRE (lines 2500 - 2860)

By reading the header record and looking for an FIC value of zero, line 2505 checks to see if the file is empty. There are two types of inquiries — BROWSE and DIRECT. BROWSE allows the user to start anywhere in the file and call records up in sequence; the user is also given the option of changing or deleting any record from the BROWSE option. DIRECT requires the user to know the full key; unless it finds a match on the exact key, it will display a 'not found' message.

Line 2509 asks the user which type of Inquiry is to be performed. If the user replies "D" for DIRECT, lines 2600 through 2630 ask the user for the key, find the record, and display the data. When the user is done viewing the record, any key can be depressed to return to the main menu.

If the user selects the BROWSE option, processing continues at line 2750, which asks the user to enter a part or whole of the last name. This tells the program where to begin the BROWSE.

Line 2820 displays the record. Line 2830 asks the user to change, delete, look at the next record in the chain, or return to the main menu (quit). To continue, lines 2836 through 2850 preprare to get the next record, first making sure that the end of the chain has not been reached (NXT(OLD) = 0). If it hasn't, then the next record is obtained by a GOSUB 9015 (line 2850). The BROWSE cycle is continued by looping back to line 2810.

QUIT (line 150)

Quit makes sure the file is properly closed so any data in the buffer is written to the file before processing ends.

Program Hints

LINK-PROG is an example of how to deal with linked-lists. It provides you with all the basic processing components of linked-lists, but is not designed as an end in itself. It is simply an example of an application of linked-lists.

Note that as the linked-list gets longer the processing begins to slow down. To access the last record in the chain all subsequent records must be read first. So linked-lists are best used where the lists are relatively short. An example of this is the catalog sectors that Apple DOS use to store file description entries. Also, linked-lists are most powerful when they are built on top of each other in a hierarchy, so that there are subchains and sub-subchains. This structure makes it possible to build large and complicated data-bases.

LINK-CREATE

```
Ø  HOME
1  INPUT "ENTER NUMBER OF RECORDS ";SZ
5 CD$ =  CHR$ (4)
1Ø  ONERR  GOTO 2Ø
15  PRINT CD$"DELETE LINK-FILE"
2Ø  PRINT CD$"OPEN LINK-FILE,L8Ø"
22  PRINT CD$"WRITE LINK-FILE,RØ"
    : PRINT Ø: PRINT 1
25  FOR J = 1 TO SZ
3Ø  PRINT CD$"WRITE LINK-FILE,R";J
4Ø  PRINT A$: PRINT A$: PRINT A$:
      PRINT A$: PRINT A$: PRINT A
    $: PRINT A$
45  IF J = SZ THEN  PRINT Ø: GOTO 5Ø
47  PRINT J + 1
5Ø  NEXT
6Ø  PRINT CD$"CLOSE LINK-FILE"
```

(LINK-PROG Listing on next page)

```
10  DIM FIRST$(2): DIM LAST$(2): DIM
     SRT$(2): DIM CITY$(2): DIM S
     T$(2): DIM ZIP$(2): DIM TEL$
     (2): DIM NXT(2)
20 CD$ =  CHR$ (4):NW = 1:WR = 2
30  PRINT CD$"OPEN LINK-FILE,L80"
100  HOME : HTAB 4: VTAB 2: PRINT
     "A(DD) C(HANGE) D(ELETE) Q(UIT)";
110  HTAB 4: VTAB 3: PRINT "I(NQU
     IRE) ";: GET A$
120  IF A$ = "A" THEN  GOTO 1000
130  IF A$ = "C" THEN  GOTO 2000
140  IF A$ = "D" THEN  GOTO 3000
145  IF A$ = "I" THEN  GOTO 2500
150  IF A$ = "Q" THEN  PRINT CD$:
      PRINT CD$;"CLOSE LINK-FILE": END
160  GOTO 100
1000  REM *** ADD RECORDS ***
1005  GOSUB 4000: REM  GET HEADER
      RECORD
1010  IF NA = 0 THEN  INVERSE : HOME
     : PRINT : VTAB 10: PRINT "FI
     LE FULL": NORMAL : FOR K = 1
      TO 2000: NEXT K: GOTO 100
1015  HOME : GOSUB 6000: REM GET
     INPUTS
1020 NR = NA:R = NA: GOSUB 5000: REM
     GET NEW RECORD
1030 NA = NXT(OLD)
1040  IF FIC = 0 THEN NXT(NW) = 0
     :FIC = NR: GOTO 1500
1050 R = FIC: REM GET FIRST DATA
     RECORD
1055  GOSUB 5000: REM GET DATA RECORD
1060  IF  NOT (LAST$(NW) + FIRST$
     (NW) < LAST$(OLD) + FIRST$(O
     LD)) GOTO 1070
1061 NXT(NW) = R
1062  IF R = FIC THEN FIC = NR: GOTO 1500
1064 R = LR: GOSUB 5000:NXT(OLD) =
     NR:T = WR:F = OLD: GOSUB 800
     0: GOSUB 7000: GOTO 1500
1070  IF NXT(OLD) = 0 THEN NXT(OL
     D) = NR:F = OLD:T = WR: GOSUB
     8000: GOSUB 7000:NXT(NW) = 0
     : GOTO 1500
1075 LR = R: REM  SAVE PREV. REC. NO.
1080 R = NXT(OLD): GOTO 1055: REM
      GET NEXT DATA RECORD
1500  GOSUB 4500: REM  WRITE 1ST REC.
1510 R = NR:F = NW:T = WR: GOSUB
     8000: GOSUB 7000
1520  GOTO 100
2000  REM
2001  REM *** CHANGE RECORD ****
2002  REM
2004  GOSUB 4000: REM GET HEADER
2005  IF FIC = 0 THEN  HOME : VTAB
     5: HTAB 4: PRINT "THERE ARE
     NO RECORDS TO CHANGE ": FOR
     K = 1 TO 2000: NEXT K: GOTO 100
2010  HOME : HTAB 4: VTAB 4: PRINT
     "DO YOU WISH TO CHANGE THE NAME ?"
 015  HTAB 4: VTAB 5: PRINT "ENTE
     R Y(ES) OR N(O) ";: GET CN$
2017  IF CN$ = "N" GOTO 2050
2020  IF CN$ < > "Y" GOTO 2015
2030  GOSUB 3030: REM  DELETE REC.
2031 F = NW:T = OLD: GOSUB 8000: REM
     RESTORE DELETED RECORDS DATA
2032  HOME : HTAB 4: VTAB 5: INPUT
     "ENTER NEW FIRST NAME ";FIRS
     T$(OLD)
2034  IF FIRST$(OLD) = "" GOTO 2032
2036  HTAB 4: VTAB 6: INPUT "ENTE
     R NEW LAST NAME  ";LAST$(OLD)
2038  IF LAST$(OLD) = "" GOTO 2036
2040  GOTO 2070: REM  GET OTHER I
     NPUTS
2050  HOME : GOSUB 6000: REM GET
     INPUTS
2060  GOSUB 9000: REM  FIND RECORD
2070  IF RC > 0 THEN  GOSUB 9075:
     GOTO 100: REM (RC=1 OR 2) =
     NOT FOUND
2080  GOSUB 9500: REM  DISPLAY REC.
2100  HTAB 4: VTAB 5: PRINT "DO Y
     OU WISH TO CHANGE THE STREET
     ? ";: GET CH$
2110  IF CH$ = "N" GOTO 2140
2120  IF CH$ < > "Y" GOTO 2100
2130  HTAB 4: VTAB 5: CALL  - 868
     : INPUT "ENTER STREET ";SRT$
     (OLD)
2135  IF SRT$(OLD) = "" GOTO 2130
2137  GOSUB 9500: REM REDISPLAY REC.
2140  HTAB 4: VTAB 5: CALL  - 868
     : PRINT "DO YOU WISH TO CHAN
     GE THE CITY? ";: GET CH$
2150  IF CH$ = "N" GOTO 2180
2160  IF CH$ < > "Y" GOTO 2140
2170  HTAB 4: VTAB 5: CALL  - 868
     : INPUT "ENTER CITY    ";CITY
     $(OLD)
2175  IF CITY$(OLD) = "" GOTO 2170
2177  GOSUB 9500: REM REDISPLAY REC.
2180  HTAB 4: VTAB 5: PRINT "DO Y
     OU WISH TO CHANGE THE STATE?
     ";: GET CH$
2190  IF CH$ = "N" GOTO 2220
2200  IF CH$ < > "Y" GOTO 2180
2210  HTAB 4: VTAB 5: CALL  - 868
     : INPUT "ENTER STATE   ";ST$(
     OLD)
2215  IF ST$(OLD) = "" GOTO 2210
2217  GOSUB 9500: REM REDISPLAY R
     ECORD
2220  HTAB 4: VTAB 5: CALL  - 868
     : PRINT "DO YOU WISH TO CHAN
     GE THE ZIP?";: GET CH$
2230  IF CH$ = "N" GOTO 2260
2240  IF CH$ < > "Y" GOTO 2220
2250  HTAB 4: VTAB 5: CALL  - 868
     : INPUT "ENTER ZIP     ";ZIP$
     (OLD)
2255  IF ZIP$(OLD) = "" GOTO 2250
2257  GOSUB 9500: REM REDISPLAY REC.
2260  HTAB 4: VTAB 5: PRINT "DO Y
     OU WISH TO CHANGE THE TEL. #
     ?";: GET CH$
2270  IF CH$ = "N" GOTO 2300
2280  IF CH$ < > "Y" GOTO 2260
2290  HTAB 4: VTAB 5: CALL  - 868
     : INPUT "ENTER TEL. # ";TEL$
     (OLD)
2295  IF TEL$(OLD) = "" GOTO 2290
2297  GOSUB 9500: REM REDISPLAY REC.
2300  IF CN$ = "Y" THEN CN$ = "":
     F = OLD:T = NW: GOSUB 8000: GOTO
     1020: REM RE-ADD RECORD FOR
     NAME CHANGE
2310 F = OLD:T = WR: GOSUB 8000: GOSUB
     7000: REM REWRITE RECORD
2320  GOTO 100
2500  REM
2501  REM ***** INQUIRY *****
2502  REM
2505  GOSUB 4000: IF FIC = 0 THEN
     HOME : HTAB 4: VTAB 5: PRINT
     "THERE ARE NO RECORDS TO INQ
     UIRE UPON": FOR K = 1 TO 200
     0: NEXT K: GOTO 100
2509  HOME : HTAB 5: VTAB 4: PRINT
     "WHICH TYPE OF INQUIRY? "
2510  HTAB 5: VTAB 5: PRINT "D(IR
     ECT) OR B(ROWSE) ";: GET B$
2520  IF B$ = "B" GOTO 2750
2550  IF B$ < > "D" GOTO 2510
2600  HOME : GOSUB 6000: REM  GET
     KEY
2610  GOSUB 9000: REM FIND RECORD
2612  IF RC > 0 THEN  GOSUB 9075:
     GOTO 100: REM IF NOT FOUND
     DISPLAY MESG. AND RETURN
2620  HOME : GOSUB 9500: REM  DIS
     PLAY RECORD
2630  HTAB 5: VTAB 23: INVERSE : PRINT
     "HIT RETURN TO CONTINUE ";: GET
     B$: NORMAL : GOTO 100
2750  HOME : HTAB 1: VTAB 4: PRINT
     "ENTER A PORTION OF (OR WHOL
     E) LAST NAME": HTAB 1: VTAB
     5: INPUT "FOR START OF BROWS
     E ";LAST$(NW):FIRST$(NW) = ""
2800  GOSUB 9000: REM  FIND REC.
2810  IF RC = 1 THEN  GOSUB 9075:
     GOTO 100: REM  DISPLAY NOT
     FOUND MESG. AND RETURN TO MA
     IN MENU
2820  GOSUB 9500: REM  DISPLAY REC.
2830  HTAB 1: VTAB 23: INVERSE : PRINT
     "ENTER N(EXT) C(HANGE) D(ELE
     TE) Q(UIT)";: GET B$: NORMAL
2831  HTAB 1: VTAB 23: CALL  - 868
2832  IF B$ = "C" THEN F = OLD:T =
     NW: GOSUB 8000: GOTO 2100
2833  IF B$ = "D" THEN F = OLD:T =
     NW: GOSUB 8000: GOTO 3045
2834  IF B$ = "Q" GOTO 100
2835  IF B$ < > "N" GOTO 2830
2836 R = NXT(OLD): REM PREPARE TO
     GET NEXT RECORD
2840  IF R = 0 THEN  HOME : HTAB
     5: VTAB 4: PRINT "THE END OF
     FILE HAS BEEN REACHED";: FOR
     K = 1 TO 2000: NEXT K: GOTO 100
2850  GOSUB 9015: REM CONTINUE BROWSE
2860  GOTO 2810
3000  REM
3001  REM  *** DELETE RECORD ****
3002  REM
3010  GOSUB 4000: REM  GET HEADER
3020  IF FIC = 0 THEN  HOME : VTAB
     5: HTAB 4: PRINT "THERE ARE
     NO RECORDS TO DELETE": FOR K
      = 1 TO 2000: NEXT K: GOTO 100
3030  HOME : GOSUB 6000: REM GET
     FIRST + LAST NAME
3040  GOSUB 9000: REM  FIND REC.
3041  IF RC = 0 GOTO 3045
3042  GOSUB 9075: REM  DISPLAY NO
     T FOUND MESSAGE
3043  IF A$ = "C" THEN  POP : REM
      IF CHANGE
3044  GOTO 100
3045 DR = R: REM  SAVE NO OF DELE
     TED RECORD
3050 F = OLD:T = NW: GOSUB 8000: REM
     SAVE DELETED RECORDS DATA
3060  IF R = FIC THEN FIC = NXT(O
     LD): GOTO 3100
3070 R = LR: GOSUB 5000: REM GET
     PREV RECORD
3080 NXT(OLD) = NXT(NW):F = OLD:T
      = WR: GOSUB 8000: GOSUB 700
     0: REM UPDATE PREV RECORD
3100 NXT(WR) = NA:NA = DR
3110 R = DR: GOSUB 7000: REM  UPD
     ATE DELETED RECORD
3120  GOSUB 4500: REM UPDATE HEADER
3125  IF A$ = "C" THEN  RETURN
3130  GOTO 100
3140  REM
3997  REM
3998  REM ***** READ HEADER ****
3999  REM
4000  PRINT CD$
4001  PRINT CD$"READ LINK-FILE,R0"
4010  INPUT FIC: INPUT NA
4020  PRINT CD$
4030  RETURN
4040  REM **********************
4500  REM
4510  REM ***** WRITE HEADER ***
4515  REM
4517  PRINT CD$
4520  PRINT CD$"WRITE LINK-FILE,R0"
4530  PRINT FIC: PRINT NA
4540  PRINT CD$
4550  RETURN
4560  REM **********************
4999  REM
5000  REM *** READ DATA REC ****
5001  REM
5005  PRINT CD$
5010  PRINT CD$"READ LINK-FILE,R";R
5020  INPUT FIRST$(OLD)
5030  INPUT LAST$(OLD)
5040  INPUT SRT$(OLD)
5050  INPUT CITY$(OLD)
5060  INPUT ST$(OLD)
5070  INPUT ZIP$(OLD)
5080  INPUT TEL$(OLD)
5090  INPUT NXT(OLD)
5100  PRINT CD$
5110  RETURN
5120  REM **********************
5997  REM
5998  REM ***** GET INPUTS *****
5999  REM
6000  HTAB 4: VTAB 4: INPUT "INPU
     T FIRST NAME ";FIRST$(NW)
6010  HTAB 4: VTAB 5: INPUT "INPU
     T LAST NAME  ";LAST$(NW)
6015  IF A$ < > "A" GOTO 6060: REM
     ONLY NEED FIRST AND LAST NAME
6020  HTAB 4: VTAB 6: INPUT "INPU
     T STREET      ";SRT$(NW)
6030  HTAB 4: VTAB 7: INPUT "INPU
     T CITY        ";CITY$(NW)
6035  HTAB 4: VTAB 8: INPUT "INPU
     T STATE       ";ST$(NW)
6040  HTAB 4: VTAB 9: INPUT "INPU
     T ZIP         ";ZIP$(NW)
6050  HTAB 4: VTAB 10: INPUT "INP
     UT TELEPHONE  ";TEL$(NW)
6060  RETURN
6070  REM
6999  REM
7000  REM *** WRITE DATA REC ***
7001  REM
7005  PRINT CD$
7010  PRINT CD$"WRITE LINK-FILE,R";
7020  PRINT FIRST$(WR)
7030  PRINT LAST$(WR)
7040  PRINT SRT$(WR)
7050  PRINT CITY$(WR)
7060  PRINT ST$(WR)
7070  PRINT ZIP$(WR)
7080  PRINT TEL$(WR)
7090  PRINT NXT(WR)
7100  PRINT CD$
7110  RETURN
7120  REM
7200  RETURN
8000  REM ****** MOVE DATA *****
8010 FIRST$(T) = FIRST$(F)
8020 LAST$(T) = LAST$(F)
8030 SRT$(T) = SRT$(F)
8040 CITY$(T) = CITY$(F)
8050 ST$(T) = ST$(F)
8060 ZIP$(T) = ZIP$(F)
8070 TEL$(T) = TEL$(F)
8080 NXT(T) = NXT(F)
8090  RETURN
8100  REM
9000  REM
9001  REM **** FIND RECORD *****
9002  REM
9010 R = FIC: REM START AT BEGINN
     ING OF FILE
9015 RC = 0: REM RESET RETURN CODE
9020  GOSUB 5000
9030  IF FIRST$(NW) + LAST$(NW) =
     FIRST$(OLD) + LAST$(OLD) THEN
      RETURN : REM  RECORD FOUND
9035  IF LAST$(NW) + FIRST$(NW) <
     LAST$(OLD) + FIRST$(OLD) THEN
     RC = 2: RETURN
9040 LR = R:R = NXT(OLD): REM REM
     ADVANCE TO NEXT RECORD
9050  IF R = 0 THEN RC = 1: RETURN
9060  GOTO 9020: REM GET NEXT REC
     ORD
9075  HOME : HTAB 4: VTAB 4: PRINT
     "RECORD NOT FOUND": FOR K =
     1 TO 2000: NEXT K: RETURN
9500  REM
9501  REM *** DISPLAY RECORD ***
9502  REM
9505 : HOME
9510  HTAB 5: VTAB 7: PRINT "FIRS
     T NAME     = ";FIRST$(OLD)
9520  HTAB 5: VTAB 8: PRINT "LAST
      NAME      = ";LAST$(OLD)
9530  HTAB 5: VTAB 9: PRINT "STRE
     ET         = ";SRT$(OLD)
9540  HTAB 5: VTAB 10: PRINT "CIT
     Y            = ";CITY$(OLD)
9550  HTAB 5: VTAB 11: PRINT "STA
     TE           = ";ST$(OLD)
9560  HTAB 5: VTAB 12: PRINT "ZIP
      CODE        = ";ZIP$(OLD)
9570  HTAB 5: VTAB 13: PRINT "TEL
     EPHONE NO. = ";TEL$(OLD)
9580  RETURN
```

SAVE 20%!!!!

Subscribe to MICRO™

Use This Postage Paid Card to Order the Next 12 Issues of MICRO and SAVE $6.00 Off Newsstand Price!

MICRO™

SAVE 20%

☐ Yes! Enter my subscription to MICRO, and send me the next 12 issues for just **$24.00.** I save **$6.00 off** the newsstand price!

Name ______
Address ______
City ______
State ______ Zip ______
I'm paying by ☐ Check ☐ MO ☐ VISA ☐ MasterCard
Card # ______ Exp.Date ______
Signature ______

I OWN A:
☐ Commodore 64 ☐ VIC-20
☐ Apple ☐ PET
☐ Atari 400 ☐ Atari 800
☐ Other: ______

For Faster Service Call:
1-800-345-8112
(In PA: 1-800-662-2444
VISA or MasterCard Only

A Feast Of Computing Ideas...

Order These Books From MICRO™

MICRO™ Books

NEW for VIC-20 Users!
Mastering Your VIC-20 With 8 BASIC Projects
A book that makes learning to program your VIC-20 fun! Contains 8 projects and programs. Games, utilities — even a VIC-20 version of "VisiCalc." All 8 programs on cassette to help you learn faster.
☐ Mastering the VIC-20 @ $23.95

NEW for OSI Users!
MICRO on the OSI
Includes Machine-Language enhancements and BASIC Aids, hardware modifications for enhanced/reversed video, programs for control code and upper/lower case entry. A valuable programming tool.
☐ Micro on the OSI @ $19.95

Best Sellers for APPLE Users!
MICRO on the APPLE
Programming aids, utilities, games, enhancements. Together Volumes 1, 2, and 3 contain over 100 programs on diskette. Fully documented and explained.
☐ 3-Volume Gift-Boxed @ $59.95
☐ Vol.1☐ Vol.2☐ Vol.3 $24.95 ea.

Please rush the MICRO Books I've checked above to:
Name ______
Address ______
City ______ State ______ Zip ______
(Allow 6-8 weeks for delivery)

I'm paying by: ☐ Check ☐ MO ☐ VISA ☐ MC
Total Enclosed: $ ______
(Add $2.00 s/h per book. MA res. add 5% tax)
Visa/MC # ______
Exp. Date: ______

MICRO™ Reader Service...

For More Information On The Advertisers In This Issue!

MICRO™ Reader Service Card

The numbers below correspond to the numbers imprinted on the advertisements in this issue of MICRO. This card valid for 90 days only.

Mail information to:
Name ______
Address ______
City ______ State ______ Zip ______

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70

March 1983

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY CARD

FIRST CLASS PERMIT NO. 60, CHELMSFORD, MA

POSTAGE WILL BE PAID BY ADDRESSEE

MICRO™

34 Chelmsford Street
P.O. Box 6502
Chelmsford, MA 01824

SAVE 20%!!!!

Subscribe to MICRO

Use This Postage Paid Card to Order the Next 12 Issues of MICRO and SAVE $6.00 Off Newsstand Price!

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY CARD

FIRST CLASS PERMIT NO. 60, CHELMSFORD, MA

POSTAGE WILL BE PAID BY ADDRESSEE

MICRO™

34 Chelmsford Street
P.O. Box 6502
Chelmsford, MA 01824

A Feast Of Computing Ideas...

Order These Books From MICRO

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY CARD

FIRST CLASS PERMIT NO. 60, CHELMSFORD, MA

POSTAGE WILL BE PAID BY ADDRESSEE

MICRO™

34 Chelmsford Street
P.O. Box 6502
Chelmsford, MA 01824

MICRO Reader Service...

For More Information On The Advertisers In This Issue!

COMPU ENE

VIC-20®

VIC-20®	**Personal Computer**	$169.95
VIC-1515	**Printer**	334.95
VIC-1530	**Datasette**	67.50
VIC-1541	**Disk Drive**	375.00
VIC-1010	**Expansion Module**	139.95
VIC-1311	**Joystick**	9.95
VIC-1312	**Game Paddles**	19.95
VIC-1600	**Telephone Modem**	99.95
VIC-1210	**VIC 3K Memory Expander Cartridge** — Plugs directly into the VIC's expansion port. Expands to 8K RAM total.	34.95
VIC-1110	**VIC 8K Memory Expander Cartridge** — 8K RAM expansion cartridge plugs directly into the VIC.	52.50
CM101	**VIC 16K Memory Expander Cartridge**	99.95
CM102	**24K Memory Expander Cartridge**	119.95
VIC-1011A	**RS232C Terminal Interface** — Provides interface between the VIC-20 and RS232 telecommunications modems. Connects to VIC's user port.	39.95

PETSPEED - Basic Compiler for Commodore — 130.00
Compile any Pet Basic program. The only optimizing compiler. Programs compiled with Petspeed run up to 40 times faster. Petspeed code is unlistable and compiled programs cannot be tampered with. No security device required for compiled programs. Available NOW for the Commodore 64.

Star Gemini 10 Printer — Call for price
Star Gemini 15 Printer — Call for price
SMD Monitor — Call for price

CARDBOARD 6 — $87.95
An expansion interface for the VIC-20. Allows expansion to 40K or accepts up to six games. May be daisy chained for more versatility.

CARDBOARD 3 — $39.95
Economy expansion interface for the VIC-20.

CARD "?" CARD/PRINT — $79.95
Universal Centronics Parallel Printer Interface for the VIC-20 or CBM-64. Use an Epson MX-80 or OKIDATA or TANDY or just about any other.

CARDETTE — $39.95
Use any standard cassette player/recorder with your VIC-20 or CBM-64.

LIGHT PEN — $29.95
A light pen with six good programs to use with your VIC-20 or CBM-64.

HOME & BUSINESS PROGRAMS For VIC-20 & C-64

CW-107A	**Home Calculation Program Pack**	$48.95
CPV-31	**Data Files** - your storage is unlimited	14.95
CPV-96	**Household Finance Package** - to keep records of all your household expenses	30.95
CPV-208	**Bar-Chart** - display your numerical data	8.95
CH	**Turtle Graphics** - learn programming	34.95
CH	**VIC Forth** - is a powerful language for BASIC programming	49.95
CH	**HES MON** - is a 6502 machine language monitor with a mini-assembler	34.95
CH	**HES Writer** - time-saving word processing tool	34.95
CH	**Encoder** - keep your personal records away from prying eyes	34.95
CT-21	**Statistics Sadistics** - statistical analysis	14.95
CT-121	**Total Time Manager 2.0** - creates personal or business schedules	15.95
CT-124	**Totl Label** - a mailing list and label program	13.95
CT-125	**Totl Text BASIC**	15.95
CT-126	**Research Assistant** - keep track of reference data	17.50
CT-140	**Totl Text Enhanced**	29.95
CM-152	**Grafix Designer** - design graphic characters	12.95
CQ-5	**Minimon** - allows you to program, load, save, or execute machine language programs	13.95
CT-3	**Order Tracker**	15.95
CT-4	**Business Inventory** - to maintain record of inventory	15.95
CS	**Home Inventory** - lists your home belongings	17.95
CS	**Check Minder** - (V-20 & 64) keep your checkbook the right way	14.95
CS	**General Ledger** - a complete general ledger	19.95
CHC-504	**HES Writer** - word processor	39.95
CHC-503	**Turtle Graphics II** - utilizes the full graphics of your 64	49.95
CHC-502	**HESMON** - machine language monitor w/mini-assembler	34.95
CHP-102	**6502 Professional Development System**	29.95
CFC	**Data Files** - a management program	27.95
CPV-327	**HESCOM** - transfers data and programs bidirectionally between VICs at three times the speed of a disk drive	40.95
CPV-328	**HESCOUNT** - monitors program execution	19.95
CHV	**HESPLOT** - Hi-res graphics subroutines	12.95
CPV-367	**Conversions** - figures, volume, length, weight, area, and velocity to all possible configurations	7.95
CC	**The Mall** - your complete mail program — Cassette	24.95
	Disk	29.95
CPV-220	**Client Tickler**	16.95
CPV-221	**Club Lister**	13.95
CPV-224	**Depreciator**	9.95
CPV-236	**Investment Analyst** - keep track of investments and investment opportunitites	12.95
CPV-251	**Present Value**	10.95
CPV-269	**Super Broker**	12.95
CPV-270	**Syndicator** - calculates whether to buy or sell	13.95
CPV-274	**Ticker Tape** - maintains investments profile	14.95
CPV-276	**Un-Word Processor** - screen editor	16.95
CPV-286	**Phone Directory** - never lose a phone number again	9.95
CS-111	**Checkbook** - home "utility" program	14.95
CPV-294	**Calendar My Appointments** - print a calendar for every month in any year.	14.95
CPV-296	**The Budgeter** - place your personal finances in order	12.95

CS1 — **QUICK BROWN FOX** — $60.50
The Word Processor of this decade!

COMMODORE SOFTWARE

VIC-1211A — **VIC-20 Super Expander** — $57.99
Everything Commodore could pack into one cartridge - 3K RAM memory expansion, high resolution graphics plotting, color, paint and sound commands. Graphic, text, multicolor and music modes. 1024x1024 dot screen plotting. All commands may be typed as new BASIC commands or accessed by hitting one of the VIC's special function keys. Includes tutorial instruction book. Excellent for all programming levels.

VIC-1212 — **Programmer's Aid Cartridge** — $45.99
More than 20 new BASIC commands help new and experienced programmers renumber, trace and edit BASIC programs. Trace any program line-by-line as it executes, pause to edit. Special KEY command lets programmers redefine function keys as BASIC commands, subroutines or new commands.

VIC-1213 — **VICMON Machine Language Monitor** — $48.99
Helps machine code programmers write fast, efficient 6502 assembly language programs. Includes one line assembler/disassembler.

NEW GAMES FOR YOUR VIC-20®

CC58	**Astroblitz** - This game is challenging, even to a VIC-MASTER! Navigate your ship carefully to avoid being hit by enemy fire.	$39.95
CC60	**Terraguard** - Speed and careful skill will enable you to once again destroy the aliens. Too slow? You're destroyed by their beam.	39.95
CC98	**Serpentine** - This game will test your patience & skill. Object - to survive long enough to lay eggs and raise your young.	39.95
CC500	**Intruder-Scrambler** - In your bomber, invade the defending scramble system, dodging rockets, to blow up enemy posts, etc.	19.95
CC101	**Choplifter** - Rescue the American hostages & return them safely to the U.S. You will encounter tanks, jets and killer satellites	39.95
CC102	**Black Hole** - Your mission is, simply, to survive! Your ship must not be hit by space objects or sucked into the Black Hole!	39.95
CC104	**Apple Panic** - Speed is required! Destroy the apple monsters by digging holes in the brick floors for them to fall into	39.95
CC65	**Video Mania** - Introducing your enemies: EVIL EYE, WALWOKER, KILLERBOX. Your only defense - throw your alien zapper!	39.95
CS1	**Flags of Nations** - A game that challenges players to identify flags of various widely-known nations of the world.	10.95
CS2	**Flags of Nations** - Second Edition - A field of 34 flags of lesser known nations of the world	10.95
CS3	**Cities and States** - A game that draws a map of a state or states and asks players to name key cities in those states.	10.95
CS4	**Cities of the World** - Deals with important cities of nations throughout the world.	10.95
CS5	**Mountains and Rivers** - Draws large geographical area maps. You identify major mountain ranges, rivers & bodies of water	10.95

NEW GAMES FOR YOUR C-64

Tank Arcade (Also for VIC-20) - Pre-determine how many hits it will take to wipe out your opponent. Then, on with the battle! Battlefield changes.	$13.95
Roadracer - Choose the type of track & a time or lap race. Use steady control at speeds of 50 to 200 miles per hour. Hit the wall & lose valuable time.	13.95
Shootout at the OK Galaxy (Also for VIC-20) - 30 alien warships have entered your war zone. Shields up? Energy level OK? Defend yourself.	19.95
Galaxy - Have you ever wanted to conquer the universe? Send your galactic fleets out to explore, solar system by solar system. From 1 to 20 players.	19.95
Bomber Attack - Ground to air warfare. You're in command of a supersonic bomber over enemy terrain. Drop all 25 bombs on key locations.	14.95
Midway Campaign - Your computer controls a huge force of Japanese ships trying to conquer Midway Island. Your only advantage is surprise.	19.95
Dnieper River Line - A fictionalized engagement between Russian & German forces in 1943. Soviet forces, controlled by the computer, seek to overrun your line and capture sufficient objectives to attain victory. Four levels of difficulty.	25.00
Tanktics - Armored combat on the Eastern front of WWII. You start outnumbered 2 to 1 but you choose your tank types before the battle.	24.50
Guns of Fort Defiance - You are the commander of a 19th artillery piece in a besieged fort. Choose type of ammo. Set the cannon's elevation, deflection.	20.00
Computer Baseball Strategy - You, the manager of the home team, test you skill against a wily and unpredictable opponent, your computer.	15.95
Lords of Karma - Like an intriguing puzzle! Decipher secrets while exploring a mythical, magical city & countryside. Avoid the lurking monsters!	20.00
North Atlantic Convoy Raider - It's the Bismarck convoy raid of 1941! The computer controls the British ships. Will you change history?	19.95
Planet Miners - Compete against others and the computer to stake valuable mining claims throughout the solar system in the year 2050.	19.95
Conflict 2500 - In 2500 AD, earth is threatened by attacking aliens with an infinite # of attack strategies with which to tease the defending player.	19.95
Nukewar - Nuclear confrontation between two hypothetical countries. Defend your country with espionage, bombers, missiles, submarines, etc.	19.95
Computer Acquire - New Second Edition! The object is to become the wealthiest person in this "business" game - hotel acquisitions & mergers.	20.00
Andromeda Conquest - Vast scale space strategy game of galactic colonizing and conquest. Strange life forms & alien technologies - exciting!	19.95
Telengard - Microcomputer Dungeon Adventure game. Time fantasy and role-playing. 50 levels of ever-more complex mazes to explore & survive!	25.00

MORE — MORE — MORE

Prices subject to change.

TO ORDER:
P.O. Box 18765
Wichita, KS 67218
(316) 684-4660

WRITE FOR FREE CATALOG

MasterCard — VISA

Personal checks accepted (Allow 3 weeks) or C.O.D. (Add $2) — *Handling charges $2.00*

VIC-20® is a registered trademark of Commodore

Hi-Res Plotting with the VIC

by Fred Wallace

A discussion of the graphics mode available on a VIC-20 equipped with the VIC 1515 graphic printer is presented, along with a BASIC subroutine that permits plotting of either a mathematical function or a user-generated array on a grid with labelled axes.

Hi-Res Plotting
requires:
VIC-20 (5K or more)
VIC-1515 or
VIC-1525 printer

Mathematical relationships are best analyzed in the form of a graph. Even an inexpensive computer like the VIC-20 can be used to produce such a graph if the computer is used with a pin-programmable dot-matrix printer.

For those who own the VIC-20, Commodore has provided a very capable printer, the VIC-1515, which has features that make plotting easy. Although most people probably will purchase the 1515 for program listings and data printout only, it comes with some very powerful graphics and formatting commands. The commands are explained thoroughly in the manual that accompanies the unit, and the program presented here uses most of them. The newer VIC-1525 printer uses standard-width paper, but is programmed similarly.

The VIC-1515 has three major operating modes. When first powered up, or after receiving a CHR$(15), it is in the standard-character mode, which permits printing of all alphanumeric and keyboard-graphics characters available on the screen. This mode is used for program listings and normal data printout. Access to the alternate set of keyboard graphics (i.e., those on the left front of the keycaps) is with a special secondary address in the OPEN command.

Double-width characters are obtained with a CHR$(14). This expands each character horizontally, reducing the number of characters per line from 80 to 40 in the process. These may be used in either the standard or alternate printer character sets.

The graphics mode is entered by transmitting a CHR$(8). In this mode you gain control of each individual "dot" in the print-line. To understand how this is done, first look at how the character modes work.

When the printer receives the code for a particular character, it first looks the code up in the built-in ROM to see what dots are required to form that character. The ROM supplies information for a 5 × 7 matrix of dots (see the upper-case "A" in figure 1) and then the printer logic leaves the sixth column blank before proceeding to the next character. By sending one 8-bit code, you can print anywhere from 0 to 35 dots.

In contrast, the graphics mode allows you to bypass the printer's character ROMs and send information to the head directly (see figure 2). Once in this mode, each 8-bit code received by the printer is no longer interpreted as a character — it is either processed as a control code if bit 7 is zero (that is, if the binary value is less than 128), or it is sent directly to the head if bit 7 is one (binary value between 128 and 255). In the latter case, bits 0 through 6 determine directly whether or not the head will place a dot on dot-lines 1 through 7 of this particular print-line. For example, in order to place a single dot in dot-line 4, the binary value required is 128 + 8, or CHR$(136). Dot-lines 3 and 4 would be darkened by sending 128 + 4 + 8, or CHR$(140). Since you must

Figure 1: 5x7 dot matrix for printer character 'A'.

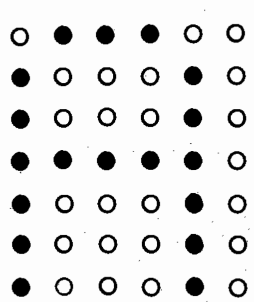
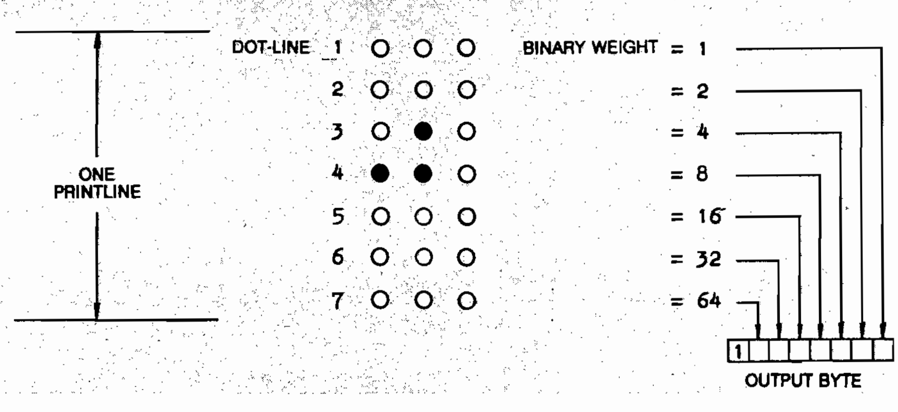


Figure 2: Pin-programmable graphics mode.



issue one such byte for each vertical column of dots (remember, there are six of these in each normal-mode character), it is obvious that you can place a dot anywhere you want it, within the area reached by the printhead on each pass. For example, to duplicate the upper-case "A" in figure 1 you would have to send CHR$ codes of 254, 137, 137, 137, 254, 128.

The printer also automatically takes care of buffer overflow in the graphics mode. It may take as many as 481 bytes of data to print one line in this mode (i.e., the CHR$(8) to get into graphics, followed by 80 × 6 = 480 data bytes), but the printer's internal buffer is only 90 bytes long. Therefore it will print partial lines, 90 bytes at a time, until it completes the whole line. This action is taken without any special programming requirement. To avoid having space left between successive lines, the linefeed length is two-thirds of that used in the character mode.

Plotting with Dots

Although the most common use for the graphics mode is to create customized characters (like the little monsters you chase all over the screen in games), the real value lies in its ability to place a dot almost anywhere on a piece of paper. In the graphics mode, approximately 98 print-lines will fit on an 11-inch-long sheet of paper — that is 98 × 7 or 686 dot-lines. You have seen that the 80-column width means 80 × 6 = 480 possible horizontal points, so on an 8 × 11 page you have available a grid of 686 (vertical) by 480 (horizontal) dot positions — high enough resolution for some very fine work.

The subroutine in listing 1 is designed to use the grid concept to create a graph of any two-dimensional relationship for which a VIC-BASIC DEF FN can be written. You need only to reserve an integer array (A%), define the function, and specify the start-value and step-value (increment) for X and Y (variables XS, XI, YS, and YI, respectively). If a title is placed in array T$, it is printed under the completed graph. The routine uses only about 1600 bytes so plenty of room remains, even on a 3.5K machine, for data-entry or tape-read routines to automate the process. For example, calling the subroutine from the program in listing 2 produced the plot shown in graph 1.

The Program

Lines 1000 to 1020 perform initial setup. B$ is the VIC-printer equivalent of a TAB function and advances the printhead to the twentieth character position. The tab is used for locating the vertical axis. C$ sets the printer to standard-character mode. D1$ and D2$ print the characters that form the vertical axis, and then put the printer in graphics mode, ready to plot.

Line 1030 begins the outer functional loop. YR is the Y-increment per print-line (7 dot-lines), and RV is the row-value of the uppermost dot-line in each print-line. Line 1040 initializes the output array to "graphic spaces"

Listing 1

```
998 REM
999 REM -------------------- PLOTTING SUBROUTINE --------------------
1000 B$=CHR$(16)+CHR$(50)+CHR$(48):C$=CHR$(15)
1010 D1$=CHR$(171)+CHR$(8):D2$=CHR$(123)+CHR$(8):SP=128
1020 OPEN1,4:PRINT#1:PRINT#1
1030 YR=YI*7:FORRN=0TO49:RV=YS+(50-RN)*YR
1040 FORCN=0TO300:A%(CN)=SP:NEXTCN
1050 IFRN-10*INT(RN/10)<>5THENPRINT#1,C$;B$;D1$;:GOTO1090
1060 CN=RV-3*YI:IFABS(CN)<0.5*YITHENCN=0
1070 PRINT#1,C$;:E$=STR$(CN):FORN=1TO(19-LEN(E$)):PRINT#1," ";:NEXTN
1080 PRINT#1,E$;B$;D2$;:CE=0
1090 FORCN=0TO299
1100 RS=RV-FNY(XS+CN*XI):IFRS<0THEN1130
1110 DR=INT(RS/YI):IFDR>6THEN1130
1120 A%(CN)=128+2↑DR:CE=CN
1130 NEXTCN
1150 CE=CE+1:A%(CE)=13
1160 FORCN=0TOCE:PRINT#1,CHR$(A%(CN));:NEXTCN
1170 NEXTRN:PRINT#1,C$;B$;CHR$(173);
1180 FORCN=0TO50:IFCN-10*INT(CN/10)<>5THENPRINT#1,CHR$(177);:GOTO1200
1190 PRINT#1,CHR$(123);
1200 NEXTCN:PRINT#1
1210 FORNN=5TO45STEP10:F$=STR$(NN+15):PRINT#1,CHR$(16);MID$(F$,2,2);
1220 XV=XS+(NN+0.5)*XI*6:IFABS(XV)<0.5*XITHENXV=0
1230 E$=STR$(XV):L=LEN(E$):IFL>10THENL=10
1240 FORN=0TOINT(5-L/2):PRINT#1," ";:NEXTN
1250 PRINT#1,E$;:NEXTNN:PRINT#1:PRINT#1
1260 PRINT#1,CHR$(14);:L=LEN(T$):IFL>40THENL=40
1270 FORN=1TOINT(23-L/2):PRINT#1," ";:NEXTN
1280 PRINT#1,T$;CHR$(15):PRINT#1:PRINT#1:CLOSE1
1290 RETURN
```

Listing 2

```
99 REM ------ A SAMPLE CURVE-GENERATING PROGRAM -----
100 DIM A%(300)
110 XS=-0.3:YS=-0.249:YI=.002:XI=.1
120 DEF FNY(X)=SIN(X)/(X+1)
130 T$="Y = SIN(X)/(X+1)"
140 GOSUB1000:END
```

Listing 3

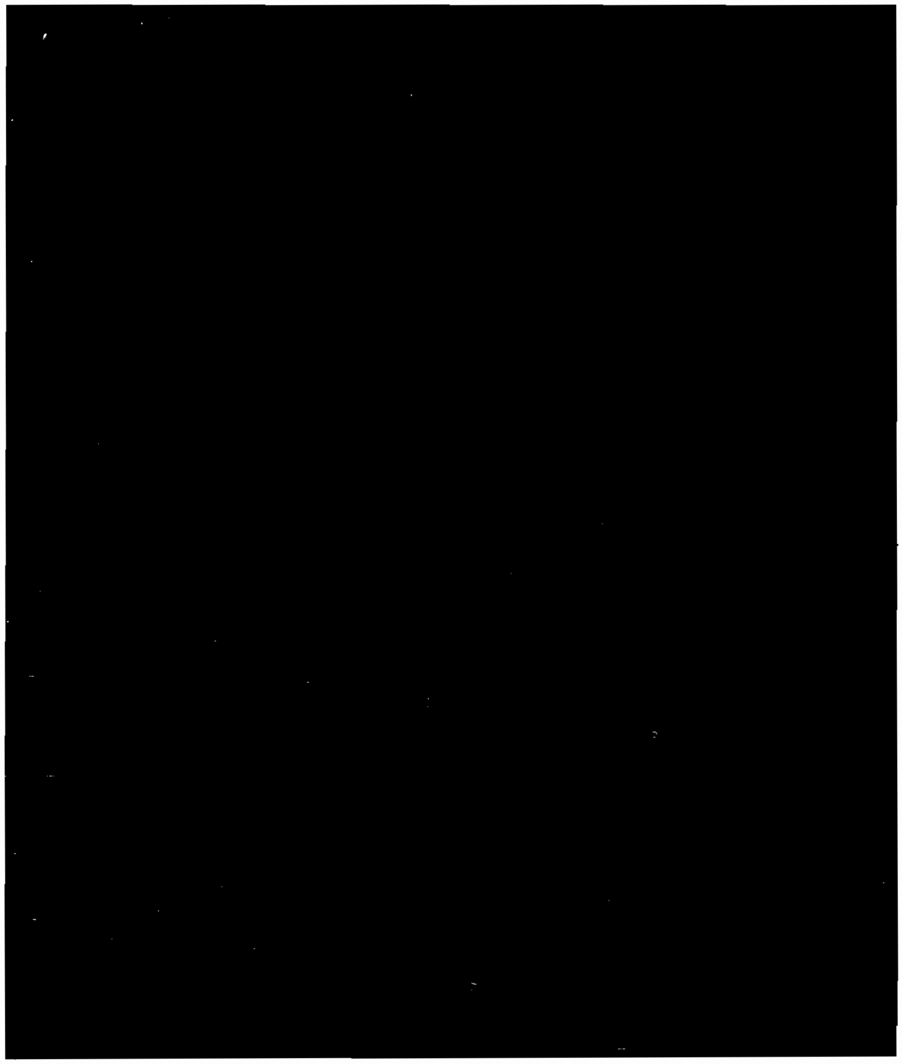
```
99 REM ------ A SAMPLE POINT-PLOTTING PROGRAM -----
100 DIM A%(300),VA(300)
110 XS=0:YS=0:YI=1:XI=1
120 VA(0)=0:FORCN=0TO299:IFCN-100*INT(CN/100)>=50THENIN=-6:GOTO125
123 IN=6
125 VA(CN+1)=VA(CN)+IN:NEXTCN
130 T$="TRIANGULAR WAVE"
140 GOSUB1000:END
        .
        .
        .
        .
        .
1100 RS=RV-VA(CN):IFRS<0THEN1130
```

(128). Line 1050 determines if it's time to place a label on the vertical axis. If it is, then lines 1060 to 1080 calculate its value, strain out "zero-residuals" such as 1.729358E-26, and right-justify the resulting string into position. Enough room is reserved so that any legal string resulting from the STR$( ) function can be printed, even those with exponents.

Lines 1090 to 1130 form the inner functional loop. For each column number (CN) you find the value of the FN Y and determine if that value lies within this print-line. If it does, make an entry in A%. Variable CE keeps track of where the last entry was made so you can truncate trailing spaces. Lines 1150 and 1160 append a carriage return and finish the line by sending A% to the printer.

When all 50 lines (that is, 350 dots vertically) have been printed, then lines 1170-1200 output the horizontal axis, complete with tic marks. Printing the labelling for this axis is somewhat more challenging than the vertical, however, since you want to *center* the number string at a given position rather than right-justify it. Lines 1210 to 1250 accomplish this, first by tabbing to the start of the allowable space for each label, then spacing over to the correct beginning point before actually outputting the string. This algorithm is less able to deal with lengthy labels because of restricted space, so you may want to do some intelligent rounding here.

Finally, lines 1260 to 1280 center the caller-supplied title in double-width characters under the graph.



Upgrading the Routine

The first complaint you will have about the above example is that it is painfully slow (figure 3 takes over 20 minutes). Much of this is due to the sheer number of bytes sent to the printer — each line requires as many as 330. Extra time is consumed with serial bus handshaking.

Astute BASIC programmers also will notice that the function FN Y is called from the innermost loop and is consequently called 50 × 300 (15000) times! Worse yet, only 300 of these calls produce unique values. This was done to save memory, as the storing of 300 floating-point numbers would use over 1500 bytes — almost as much as the routine itself! If you have an expanded VIC, you can change the code to precalculate the values and then use them directly in line 1100, but don't expect too dramatic an improvement. Interpreted BASIC is not outstanding when it has to run FOR-NEXT loops 15000 times.

You may want to try the program in listing 3. This routine eliminates the FN definition and instead loads array VA directly. Don't forget to make the accompanying change in line 1100 before running. This program, which just barely fits in an unexpanded VIC, not only runs faster but also shows how point-plotting (as opposed to function-plotting) can be implemented.

Conclusion

A VIC-20/VIC-1515 combination can produce high-resolution dot-plots comparable to dot-addressable printers available at much higher cost. With the addition of the VIC RS-232 option and a few lines of code, the package could become an intelligent graphic-output station for another micro or mainframe — one that can buffer an entire graphic and plot offline.

(*Author's note:* A few of the early-production 1515 printers do not handle buffer overflow properly in graphics mode. If your unit locks up occasionally while running these routines, contact your dealer or Commodore Customer Support.)

References

1. *VIC-20 Programmer's Reference Guide*, Commodore Business Machines, Inc. and Howard W. Sams and Company, Inc., 1982.
2. *VIC-1515 User's Manual*, Commodore Business Machines, Inc., April 1981.

Contact the author at P.O. Box 3322, Windsor, Ontario, Canada N8N 2M4.

MICRO

Circle No. 16

Circle No. 17

Print Control for APPLE Printers

by John R. Vokey and H. Cem Kaner

Automatic pagination and user-selectable margins should be part of the printer-driver routine, but are rarely included. This short machine-language subroutine provides formatted output to most output devices.

Print Control
requires:
Apple II and printer

If the area where you work on your Apple looks anything like ours, you have a mess in need of a clean-up solution. Somewhere there's a heap of program listings and, perhaps, a marginally tidier file drawer of folders holding other listings and printouts. The problem will be worse if, in a brilliant economy move, you buy a thermal printer or some other printer that uses rolled paper rather than fanfold. We made this mistake when equipping one of the five Apples that our two labs use. The rolled paper curls back into little scrolls, which are easily mutilated and defy attempts at neat filing.

There's an obvious and cheap means of bringing this problem under control. Three-hole punch the printouts and bind them in paper or cardboard binders. With some gentle persuasion (stacking about 50 pounds of books on top of the printouts for a few days usually works) even the rolled paper can be made to lay almost flat inside the binders. Unfortunately, there are a few difficulties in doing this for BASIC listings and printouts from programs that do not allow formatting of output.

The first problem occurs when you separate the pages of output (fanfold paper) or rip the rolled paper into pages. The top halves of some lines appear on the bottom of some pages while the bottom halves are at the top of the next pages. After relettering these lines by hand and three-hole punching the pages, you'll note the next problem. Some of the line numbers have been replaced by holes. This won't be noticeable, though, because the paper or cardboard strip in the binder that presses the left edge of your output flat will hide the rest of your line numbers anyway.

Automatic pagination and user-selectable margin settings are easily programmed tasks for a computer or an "intelligent" peripheral or peripheral board, but are rarely included. While some programs, such as Apple Writer, do an excellent job of formatting a page, others do not. Applesoft's LIST command, for example, has no provision for margin adjustment, except *via* text window settings for the TV screen, which are ignored by the printer. As to the printers, most rightfully expect that the computer or the interface board will tell them when to carriage return and what the margin settings should be, so there is no solution at this level either. Formatting your program listings will have to be done by either the interface or by use of a patch to BASIC. Some interface boards, such as the Grappler, do give the user control over the right and left margins of output and over the length of printed pages and the number of blank lines left between them. Most don't. For example, Grappler's closest competitor, the PKASO board, doesn't allow resetting of the left margin. One board we've used doesn't allow control of even the right margin except *via* a few DIP switch settings. This is most infuriating at times: where do you put marginal notes on program listings if there are no margins on the page? If you have a Grappler or other board that can do all the output formatting for you, read no further. Your interface does everything that our program does except for titling each page. However, if you use Apple's standard parallel output board or one of the majority of other interfaces around, and would prefer to spend a few minutes of typing to $200 to replace the board, here's a program to format your output.

Using the Program

The program is reached by a CALL command. If you load it at location 768 ($300), set PRNT = 768. Before issuing the CALL, initialize your output device (e.g., PR#1 for a printer connected to slot 1). Some printers and interfaces require a few other initialization commands, such as CTRL-I80N to allow 80-column output. A few also require that at least one carriage return be printed before the printer is considered "active" by DOS. Your printer and board manuals will give you initialization details. Do the call at the point where you normally start printing data

***Table 1:* Format Parameters**

Parameter	Label	Legal Range	Default	POKE Address and Notes
Left Margin	LM	0< =LM< =255	10	966. Blank lines forever if RM< =LM
Right Margin	RM	LM<RM< =255	70	965. Make it at least 1 less than printer allows
Page Number	PN	0 < = PN< =255	1	964
Step-over	ST	0 <= ST <?L	10	963. Blank pages forever if ST > = PL
Page Length – 1	PL – 1	ST<PL – 1< =255	65	962. PL – 1 = 0 kills pagination and title
Title		255 chars	"Page"	935 low byte, 937 high byte

instead of more printer commands. The general form of the PRNT call is

CALL PRNT,LM,RM,PN,ST,PL-1

where the parameter names (LM, etc.) are as defined in table 1. For example, with a 132-column printer you might CALL PRNT,15,117,5,10,65. Output would be indented by 15 blank spaces (LM). The right margin would be 117 characters from the left edge (not margin) of the page. The page number printed for the first page of output would be 5. Ten linefeeds would be inserted between each page of output, and each page (counting the step-over linefeeds) would be 66 lines long.

The example above contains a lot of parameters to bother with each time you call the routine. Often your parameters won't change from one call to the next, or you might be happy with certain default values (such as those in table 1). You can omit specifying a new value for any parameter either by leaving out the number but typing in the comma separating it from the next parameter, or by terminating the list early with a carriage return or colon. The values assigned to these parameters are the ones used the last time you called the routine or, if none have been specified, by the default values loaded into the appropriate locations when the program was brought into memory from the disk. For example, CALL PRNT with no parameters leaves all parameters as they were before. CALL PRNT,,80,,50 defaults the left margin, the step and the page number, and sets the right margin to 80 columns and the page length to 50 lines.

Once you issue a CALL PRNT command, the parameters (except for page number, which is updated after each page) stays constant until you turn the printer off with a PR#0 command. As long as the printer (modem, whatever) is active, you need never call the PRNT routine again. In fact, you should not call it again without first shutting it off by a PR#N command where N refers to any legal output slot. As explained below, the system will hang if PRNT is re-CALLed when it is active. If you want to change the formatting parameters while printing, just POKE new parameter values into the appropriate locations, as specified in table 1. For example, POKE 966,15 resets the left margin to 15.

Changing output parameter values on the fly can be used to produce formatted BASIC listings, with indentation and spacing used to highlight program structure (as typically done for Pascal listings). For example, if you have a long FOR...NEXT loop, or many of them nested inside others, the program structure is seen more easily if the lines inside the loops are indented a few spaces. A very simple example of formatting program output is given in listing 1. Looking at that program, it's clear that this can be a tedious business. You need to append a whole new program to your program to control the listing. However, tedious is preferable to impossible. This feature comes in most handy when you have a final draft of a program that you've written for someone else, and want to give that person a listing which is easier to read (or looks more professional) than an unformatted BASIC listing.

A final parameter, which is not specified in the parameter list, is the heading to be printed at the top of each page. As long as your page length is not set to 0, the page number will be printed. The program in listing 2 will output this as "PAGE ";PN, where PN is this page's number. The string "PAGE " is replaced easily, however. In subroutine TITLEOUT, the location of the title string is passed to STROUT, Applesoft's string output subroutine. Currently, that string is stored in locations $3C9 through $3CE (labelled TITLE in the program). If you have a different title in mind, simply define it in your program and point to it within TITLEOUT. Only the pointer at $3A7 (low byte, decimal 935) and $3A9 (937), which currently point to TITLE, need be changed.

You can change the title and the pointer in a number of ways. The one we find easiest works like this: at the very start of the program (line 1) insert a REM containing the title you want printed. For example, if you want "MY LISTING PAGE ";PN printed at the top of each page (where PN is the page number), your program's first line would be

1 REM MY LISTING PAGE

Remember to put a space after "PAGE" in the REM. You won't see it in the pro-

***Listing 1:* Print Control**

```
1  REM DOUBLESPACE BETWEEN LINES COURTESY OF APPLESOFT LIST
2  REM ELIMINABLE IF YOUR PRINTER REVERSE LINEFEEDS
10  FOR I  1 TO 10
20  REM  NESTED AT FIRST LEVEL
30  REM  SO INDENT TO SHOW IT
40  FOR J  1 TO 10
50  REM  NESTED ANOTHER LEVEL
60  REM  INDENTATION MAKES THIS CLEARER
70  NEXT J
80  REM  BACK TO THE FIRST LEVEL
90  NEXT I
99  REM  BACK TO OUTERMOST LEVEL
110  REM  NOW FORMAT THE LISTING OF THE PROGRAM
120 D$   CHR$ (4): PRINT D$;''PR1'': PRINT  CHR$ (9);''81N'':
     REM MAKE PRINTER MARGIN 1 WIDER THAN MARGIN MARGIN
130 PRNT  768: CALL PRNT: REM  DEFAULTS OK
140  LIST  - 10
150  POKE 966,15: LIST 20,40: REM INDENT
160  POKE 966,20: LIST 50,60: REM MORE INDENT
170  POKE 966,15: LIST 70,80: REM DOWN A LEVEL
180  POKE 966,10: LIST 90,99: REM END OF ACTUAL PROGRAM

]RUN
```

```
PAGE  1


1  REM DOUBLESPACE BETWEEN LINES COURTESY OF APPLESOFT LIST
2  REM ELIMINABLE IF YOUR PRINTER REVERSE LINEFEEDS
10  FOR I  1 TO 10


     20  REM  NESTED AT FIRST LEVEL
     30  REM  SO INDENT TO SHOW IT
     40  FOR J  1 TO 10


          50  REM  NESTED ANOTHER LEVEL
          60  REM  INDENTATION MAKES THIS CLEARER


     70  NEXT J
     80  REM  BACK TO THE FIRST LEVEL


90  NEXT I
99  REM  BACK TO OUTERMOST LEVEL
```

gram, but it will be printed if you include it.

REM specifies what the new title should be, but you haven't told the formatting program where to find it. The title itself starts in memory six bytes after the start of program text, as pointed to by Applesoft's pointer TXTTAB at $67,$68. For example, if TXTTAB holds $801 (the usual case), the M of MY LISTING PAGE starts at location $807. POKE 8 into location 937 ($3A9) and POKE 7 into location 935 ($3A7) and your title, instead of the default title, will be printed at the top of each page.

We should stress that the program is handy for more than just producing listings. Often we print long tables of data that run for more than a page. Using this routine we can indent the first column of data automatically and leave blank lines between pages. Further, we can automatically print the headings for each column of output at the top of each page by making that our "title." For example,

```
1 REM PARAMETER   LABEL   LEGAL
RANGE   DEFAULT   POKE ADDRESS
& NOTES PAGE
```

would be used to print the column headers at the top of each page of output for table 1.

Notes on the Program Itself

Two things in the program are likely to puzzle anyone attempting to understand the code. The first is a BIT RTS1 command used at line 76. The second is our technique for intercepting data sent to the printer.

The BIT RTS1 command is a trick that we ran across in the old *Apple II Reference Manual* (the red one — now out of print). Control characters are not printed on the screen or on the printer, so if the program counts these as having been printed it will force a carriage return before the true right margin is reached. ASCII control codes are in the range $00 through $1F (high bit clear) or $80 through $9F (high bit set). To detect the codes in a straightforward way would require three or more compare-and-branch instructions, which would make our routine too long to fit on page 3 of memory. Note, however, that the ASCII codes are distinguished by having the second and third highest bits clear, whether the high bit is set or not. Therefore, if they were ANDed with #$60, the result would be $00. This is true for all control characters and no others. Thus, if you store the character in the accumulator, AND it with #$60, and test the zero flag, you have a control character when Z = 1. The only problem is that the AND wipes out the accumulator in the process. You need to keep the character to determine if it's the code for a carriage return, in which case tabbing to the left margin and a check on page length must be done. The BIT command performs an AND between the accumulator and the contents of a memory location, setting the right flags but leaving the accumulator alone. So if you can find a memory location holding #$60 you're in business. Since #$60 is the machine code for RTS, BIT RTS1 does exactly this job. From there you branch around the line updating the position of the print head if you do have a control character.

To intercept the data, your routine must count each character printed to tell when the right margin is reached. At that point, it breaks the line, carriage returns, and tabs (inserts blanks in the printout) to the left margin. The routine also needs to detect carriage returns sent from BASIC in order to tab over for the next line after these, and to check if a new page should be started. To accomplish this, at some point you have to pass every character being sent to the printer through your routine, counting non-control characters, carriage returns, etc. To do this, you must find out where the subroutine controlling output to the printer, modem, TV (or whatever device is active), is located in memory. After finding it, you can either direct it to pass control to your program after it's done printing the character, which is rather tricky if that routine is in ROM and is thus unmodifiable, or you can find out as well what routine calls this output routine, modify that one instead, and then pass characters to the device handler after you've looked at them.

The second approach, that of grabbing the characters before they're printed, is much simpler, due to careful thinking about output flexibility on the part of Apple's programmers. While most device handlers are stored in ROM or EPROM chips, all output is directed to these handlers through a pointer in RAM called CSW, at locations $36 and $37. On an Apple that uses only cassette tape for mass storage, CSW always holds the address of the current output device handler and all PRINTing is done by checking CSW and sending character codes to the address it contains. All you need to do, then, is to read CSW, store the address it holds somewhere in your program, and then point CSW at your routine. Send the data to the true output device on line 75 of the program, at JSR DOCHAR, where the address after JSR is modified after each CALL to point to the correct routine. (Life is more complex if you use disks.)

Many DOS commands are issued using PRINT commands. Thus, when a program is running, you can't say simply "PR#1". You have to say PRINT CHR$(4);"PR#1". The printed CTRL-D (CHR$(4)) is there to get DOS's attention. This is true also for commands to open and close files, etc. To find these CTRL-Ds, DOS has to grab each character as it goes by and check if it is a CTRL-D. How does it do that? It points CSW at itself, stores the true output device-handler address in a safe place inside the DOS code, and passes characters to the device when it's through with them. DOS is very protective about the contents of CSW. It's easy enough to change CSW, but if you do just about anything after that (turning your head often seems adequate), you'll discover that DOS changed it back. There is no way to grab the output before DOS gets it without completely rewriting the operating system. Instead, what you have to do is to fool DOS into thinking that your routine is the correct device-handling routine and let it pass the data to you. Then, as before, your routine can send it to the real device.

The first part is easy. If you change CSW to point to your output checking routine (called HOOK) at $34D, then when DOS takes back control of CSW (using a subroutine called DOSCON, for DOS CONnect), it stores $34D in the place it reserves to hold the "true" CSW value, and outputs all characters to HOOK thereafter. So far so good, but where does HOOK send them? If it uses the CSW contents as a guide, it sends them back to DOS, which sends them to HOOK, which sends them to DOS; and you've just created a lovely model of a bureaucracy at work. Nothing ever gets printed this way. Instead, you have to find out where DOS has hidden the true device-handler address and, before allowing DOS to overwrite that with HOOK's address, you have to save it in HOOK. This is what the CONDOS section of the program does.

Every time you CALL PRNT, CONDOS is executed. The first thing it does is to find DOS itself. This changes from machine to machine, depending on the amount of memory available, but CSW (almost) always points to the right place in DOS. Now it has to find that internal DOS equivalent to CSW. Unfortunately, that's not always held in the same place, even relative to the address you've found, especially (we understand) with some customized versions of DOS being used privately. What does seem to stay constant, though, is a pair of locations $26 (38) bytes past the start of the DOS intercept routine, which holds the address of the address of the device-handling routine. If you read the address held there, you can find the right

CSW replacement (finally). Having found it, store its contents in HOOK (see lines 58 and 61 of the listing), and HOOK now sends data to the routine that actually prints it (or at least to the next intercept routine, which will pass the data on eventually).

CONDOS is how you find out where to send the data. It's also the basis for our caution against CALLing PRNT a second time. If DOS is pointing to HOOK already, the next CALL to PRNT gets CONDOS working again and it will dump HOOK's address into HOOK as the place to send data to be printed. So HOOK will JSR HOOK will JSR HOOK will.... Don't CALL PRNT twice without issuing a PR#N (N holding a slot number) between. PR#N resets the DOS CSW pointer, so the next CALL to PRNT will find a real I/O handler address in this place, as it should. Note that PR#N shuts HOOK off by rewriting DOS's CSW address. If accidentally you issue two CALL PRNTs in a row, RESET will get you out of this infinite loop.

For more information on the workings of DOS, see Worth and Lechner's *Beneath Apple DOS*. Also, note that if you do not use disks, this runaround through DOS is unnecessary in your program (and won't work). Rewrite CONDOS simply to point HOOK at the address in CSW and to point CSW at HOOK. With the extra room gained in the routine, you can insert a test in the initializing routine (where CONDOS is) to leave HOOK's print address alone if CSW is pointing to HOOK already. We'd love to build that test in for the DOS version as well, but PRNT is one of a number of programs that we swap in and out of memory as needed, all running from $300 up. To fit PRNT into $300 to $3CF, we had to leave out this error test.

Printer Idiosynchracies and Other Notes

Some printers and/or cards won't linefeed in response to a simple "PRINT" command. If nothing has been printed on a line, then PRINT: PRINT causes nothing, rather than two carriage returns on the printer. The Apple Parallel I/O card connected to a Centronics 737 gives us this problem. We don't know whether this is a defect in the card, in Apple's firmware, or in the 737. (Disconnecting the printer from the card for testing is an inconvenience for reasons not germaine to this article.) The problem exhibits itself in erroneous pagination of output from DOS Tool Kit's Editor/Assembler, in some BASIC programs that issue "PRINT" commands to insert linefeeds, and in erroneous pagination by our program. For our program the solution is simple: only one truly blank line is ever printed per page, so the line count is off by only one. Instead of specifying your page length less 1 for PL, specify the exact page length (e.g., 66 instead of 65) and all will work well.

Some interface boards do not connect themselves; i.e., do not result in the JSR DOSCON, which stores their address in DOS until after the first PRINT command after PR#N has been executed. Others do connect themselves by issuing their own "PRINT", with or without printer-specific default initialization commands. If you issue a CALL PRNT right after a PR#1 and your system hangs, then your printer, modem, or whatever is not yet "con-

***Listing 2:* Print Control Demo**

```
                1          TTL 'VOKEY AND KANER, 1982'
                2  ************************************************
                3  *                                              *
                4  *                PRINT CONTROL                 *
                5  *                                              *
                6  *     A SUBROUTINE TO CONTROL OUTPUT TO A      *
                7  *     PRINTER OR VIDEO SCREEN IN APPLESOFT     *
                8  *                                              *
                9  *                     BY                       *
               10  *               JOHN R. VOKEY                  *
               11  *                     &                        *
               12  *                H. CEM KANER                  *
               13  *             MCMASTER UNIVERSITY              *
               14  *                                              *
               15  ************************************************
               16  *
               17  * CHARACTERS AND STANDARD ADDRESSES
               18  *
               19           ORG $300        * CHANGE ONLY THIS TO RELOCATE
               20  COMMA    EQU $2C         * COMMA CHARACTER, HIGH BIT OFF
               21  CR       EQU $8D         * CARRIAGE RETURN CHARACTER
               22  CSWL     EPZ $36         * APPLESOFT OUTPUT HOOK
               23  LINNUM   EPZ $50         * WE USE THIS AS A TEMP LOCATION
               24  CHRGOT   EPZ $B7         * FETCH CHARACTER AT TEXT POINTER
               25  DOSCON   EQU $3EA        * DOS CONNECT ROUTINE
               26  STROUT   EQU $DB3A       * PRINT A STRING
               27  CHKCOM   EQU $DEBE       * CHECK FOR COMMA AT TXTPTR
               28  GETBYT   EQU $E6F8       * GET BYTE OF DATA. STORE IN X
               29  LINPRT   EQU $ED24       * PRINT A,X
               30  PRBL2    EQU $F94A       * PRINT 'X' SPACES
               31  CROUT    EQU $FD8E       * PRINT A CARRIAGE RETURN
               32  DOCHAR   EQU $FDF0       * (ADR OF COUT1).THIS IS A DUMMY
               33  *     ADDRESS.  ACTUAL OUTPUT ADR IS STORED 2 BYTES PAST
               34  *     'HOOK' AT RUN-TIME, OVERWRITING DOCHAR ADDRESS.
               35  *
0300 A9 04     36  PRNTCTRL LDA 4          * 5 PARAMS (Ø TO 4). COUNT THEM
0302 85 50     37           STA LINNUM      * IN LINNUM AS GETBYT USES ALL REGS
0304 20 B7 00  38  GETVAL   JSR CHRGOT      * RECOVER LAST CHR IN LIST.
0307 F0 10     39           BEQ NEXTPAR     * DO NEXT PARAMETER
0309 20 BE DE  40           JSR CHKCOM      * MUST BE COMMA IF NOT EOL
030C C9 2C     41           CMP COMMA      * NEXT CHR ALSO COMMA?
030E F0 09     42           BEQ NEXTPAR     * YES, SKIP TO NEXT PARAMETER.
0310 20 F8 E6  43           JSR GETBYT      * NO, STORE NEW VALUE IN X.
0313 A4 50     44           LDY LINNUM      * CURRENT PARAMETER NUMBER
0315 8A        45           TXA             * CURRENT PARAMETER VALUE TO A
0316 99 C2 03  46           STA PAGELEN,Y   * SAVE IT.
0319 C6 50     47  NEXTPAR  DEC LINNUM      * CYCLE TILL 5 DONE
031B 10 E7     48           BPL GETVAL
031D           49  *
031D A0 26     50  CONDOS   LDY $26        * WHAT IS OUTPUT DEVICE ADDRESS?
031F B1 36     51           LDA (CSWL),Y    * CSWL SHOULD HOLD IT, BUT DOESN'T
0321 85 50     52           STA LINNUM      * POINTS TO DOS INSTEAD. $26 BYTES
0323 C8        53           INY             * LATER FIND ADDRESS OF ADDRESS OF
0324 B1 36     54           LDA (CSWL),Y    * DEVICE, I.E. 'TRUE' CSWL VALUE.
0326 85 51     55           STA LINNUM1    * SAVE THIS IN LINNUM, TO GET
0328 A0 00     56           LDY Ø          * THE ADDRESS ITSELF.
032A B1 50     57           LDA (LINNUM),Y * AT LAST.
032C 8D 4F 03  58           STA HOOK2      * AND SAVE IT HERE.  NOTE THE
032F C8        59           INY             * SELF-MODIFYING CODE.  WE HAVE
0330 B1 50     60           LDA (LINNUM),Y * A JSR AT HOOK1 SO THIS IS
0332 8D 50 03  61           STA HOOK3      * JSR TO TRUE OUTPUT ROUTINE.
0335 A9 4D     62           LDA HOOK       * FINALLY POINT DOS' CSWL ADDRESS
0337 85 36     63           STA CSWL        * AT THE OUTPUT INTERCEPT, WHICH
```

Listing 2 *(Continued)*

```
0339 A9 03          LDA /HOOK       * STARTS AT HOOK BELOW, SO WE CAN
033B 85 37          STA CSWL1      * FORMAT OUTPUT BEFORE SENDING IT.
033D 20 EA 03       JSR DOSCON      * FEED INTERCEPT ADDRESS TO DOS.
0340             *
0340 AD C3 03    INIT     LDA STEP        * GET STEP OVER
0343 8D C8 03       STA LINCOUNT    * SO PAGES WILL ALWAYS START HERE.
0346 20 8E FD       JSR CROUT       * CLEAR AND TAB
0349 48             PHA             * DUMMY CHARACTER
034A 4C A2 03       JMP TITLEOUT    * PRINT PAGE  AND GOODBYE.
034D             *
034D 48          HOOK     PHA             * SAVE CHARACTER TO GO
034E 20 F0 FD       JSR DOCHAR      * PRINT IT. REM DOCHAR IS DUMMY
0351 2C 69 03       BIT RTS1        * THIS TESTS IF CTRL CHAR IN 'A'
0354 F0 03          BEQ PRINT1      * IF SO, DON'T INCREMENT CURPOS
0356 EE C7 03       INC CURPOS      * NOT CTRL CHR. PRINT MOVES CURSOR
0359 C9 8D       PRINT1   CMP CR         * GOT A CARRIAGE RETURN?
035B F0 1C          BEQ PAGETEST    * YES, ADJUST AND TAB.
035D AD C7 03    MARGINTE LDA CURPOS      * NOT CR. CHECK RIGHT MARGIN
0360 CD C5 03       CMP RIGHT       * PAST RIGHT MARGIN?
0363 90 03          BCC OUT1        * NO, EXIT.
0365 20 8E FD    OUT2     JSR CROUT       * DO CARRIAGE RETURN, THEN EXIT
0368 68          OUT1     PLA             * RECOVER OUTPUT CHARACTER
0369 60          RTS1     RTS             * RETURN TO CALLER.
036A             *
036A 8A          TAB      TXA             * SAVE X
036B 48             PHA             * ON THE STACK
036C AE C6 03       LDX LEFT        * GET LEFT MARGIN INDENTATION
036F F0 03          BEQ DONETAB     * IF Ø, NO NEED TO TAB.
0371 20 4A F9       JSR PRBL2       * PRINT 'X' BLANKS
0374 68          DONETAB  PLA             * RECOVER X
0375 AA             TAX
0376 4C 68 03       JMP OUT1        * AND EXIT.
0379             *
0379 A9 00       PAGETEST LDA Ø          * COME HERE AFTER CARR.RETURN
037B 8D C7 03       STA CURPOS      * CLEAR CURSOR POSITION
037E AD C2 03       LDA PAGELEN     * RECOVER PAGE LENGTH
0381 F0 E7          BEQ TAB         * INFINITE PAGE LENGTH
0383 EE C8 03       INC LINCOUNT    * ADVANCE LINE COUNT
0386 CD C8 03       CMP LINCOUNT    * PAGELEN > LINECOUNT?
0389 B0 DF          BCS TAB         * IF YES, DO TAB AND DONE.
038B             *
038B A9 00       STEPOVER LDA Ø          * SKIP LINES TO GET TO NEXT PAGE
038D 8D C8 03       STA LINCOUNT    * CLEAR LINE COUNTER
0390 AD C3 03       LDA STEP        * GET STEPOVER
0393 F0 0D          BEQ TITLEOUT    * NO STEPOVER, NO HEADING
0395 48             PHA             * SAVE STEP
0396 20 8E FD    STEPLOOP JSR CROUT       * CARRIAGE RETURN
0399 CE C3 03       DEC STEP        * DO 'STEP' TIMES
039C D0 F8          BNE STEPLOOP
039E 68             PLA             * RECOVER STEPOVER
039F 8D C3 03       STA STEP        * AND RESTORE IT.
03A2             *
03A2 98          TITLEOUT TYA             * SAVE THE REGISTERS
03A3 48             PHA             * ON THE STACK.  'A' ALREADY
03A4 8A             TXA             * THERE, SO NO NEED TO RESAVE IT
03A5 48             PHA
03A6 A9 C9          LDA TITLE      * GET LBYTE OF TITLE
03A8 A0 03          LDY /TITLE      * GOT HIGH BYTE
03AA 20 3A DB       JSR STROUT      * PRINT THE TITLE
03AD AE C4 03       LDX PAGENUM     * GET PAGE NUMBER
03B0 EE C4 03       INC PAGENUM     * AND ADVANCE IT FOR NEXT TIME
03B3 A9 00          LDA Ø          * HIGH BYTE OF PAGE(SO MAX255
03B5 20 24 ED       JSR LINPRT      * PRINT IT
03B8 20 8E FD       JSR CROUT       * CARRIAGE RETURN
03BB 68             PLA             * RECOVER REGISTERS
03BC AA             TAX             * RESTORE X
03BD 68             PLA
03BE A8             TAY             * RESTORED Y
03BF 4C 65 03       JMP OUT2        * EXIT WITH CARRIAGE RETURN
03C2             *
03C2             * PARAMETER LOCATIONS AND DEFAULTS
03C2 41          PAGELEN  DFS 1,65        * 66 LINES PER PAGE
03C3 0A          STEP     DFS 1,1Ø        * 1Ø LINES BETWEEN PAGES
03C4 01          PAGENUM  DFS 1,1         * START AT PAGE 1
03C5 46          RIGHT    DFS 1,7Ø        * RIGHT MARGIN  7Ø

03C6 0A          LEFT     DFS 1,1Ø        * LEFT MARGIN  1Ø
03C7 00          CURPOS   DFS 1,Ø         * RT EDGE OF SCREEN OR PRINTER
03C8 00          LINCOUNT DFS 1,Ø         * NO LINES YET PRINTED
03C9             *MSB OFF   USUAL FOR APPLESOFT STRINGS
03C9 50 41 47    TITLE    ASC 'PAGE  '
03CC 45 20 20
03CF 00                   HEX ØØ          * TERMINATE BYTE FOR TITLE
03D0             END      END
```

nected." Its address is still in CSW, where you expect to find DOS's address instead. In this case, RESET unhangs the system. Simply put a PRINT command, or PRINT " ", after your PR#1, before CALLing our routine.

Our routine assumes that Applesoft, not Integer BASIC, is the currently active BASIC in the system. It uses Applesoft internal routines, so it will not work when Integer BASIC is active. To use it to print out Monitor disassembly listings, type FP if you're in Integer BASIC, then CALL −151 and you can use the routine from the Monitor.

You can use our routine to format the output of many otherwise print-inflexible utility programs, including some assemblers. In most cases it has to be reassembled to reside at an address that doesn't conflict with the utility's memory use. All addresses are defined relative to the origin in this program, so reassembly with the origin changed will take care of this completely. Other programs, unfortunately, will not allow use of this formatter no matter where located. For example, if you want to three-hole punch the output from Apple's DOS Tool Kit Assembler without punching out the addresses printed at the left margin, buy a Grappler or similar interface board for your printer. Our best efforts to date have not succeeded in convincing this program to talk to ours. Similarly, the DOS Tool Kit Assembler won't allow output to printers connected to the game I/O, which is what we drive our Teletypes through. Nor will it allow us to vector output to printers connected to a communications card through a linefeed insertion program, which we use to drive DECWriter printers (which don't linefeed after carriage returns unless they're told to). Suggestions for fixing the Apple Assembler code to allow us to print to these and other devices would be most welcome.

John R. Vokey is an assistant professor of psychology at the University of Lethbridge in Alberta, Canada. H. Cem Kaner is at McMaster University in Hamilton, Ontario, Canada. Together they have published articles on the Apple II in *BYTE*, *Nibble*, and *Compute!* This is their first publication in MICRO. You can contact the authors at the Department of Psychology, University of Lethbridge, Lethbridge, Alberta, Canada T1K 3M4.

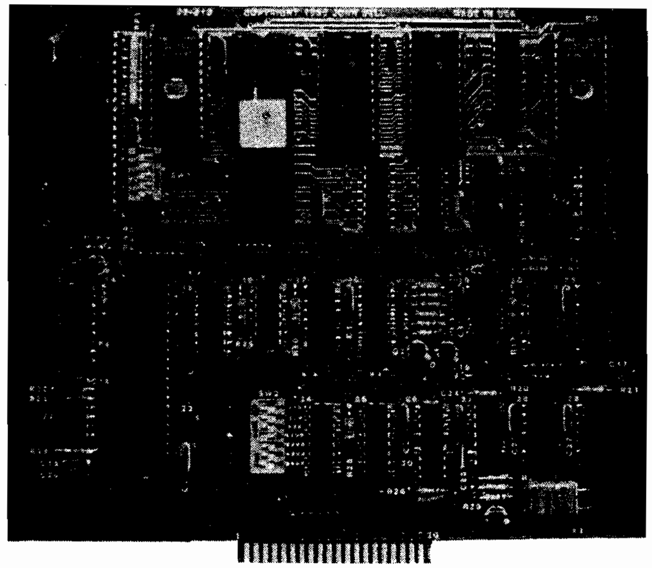
MICRO

VIDEO TERMINAL BOARD 82-018

This is a complete stand alone Video Terminal board. All that is needed besides this board is a parallel ASCII keyboard, standard NTSC monitor, and a power supply. It displays 80 columns by 25 lines of UPPER and lower case characters. Data is transferred by RS232 at rates of 110 baud to 9600 baud — switch selectable. The UART is controlled (parity etc.) by a 5 pos. dip switch.

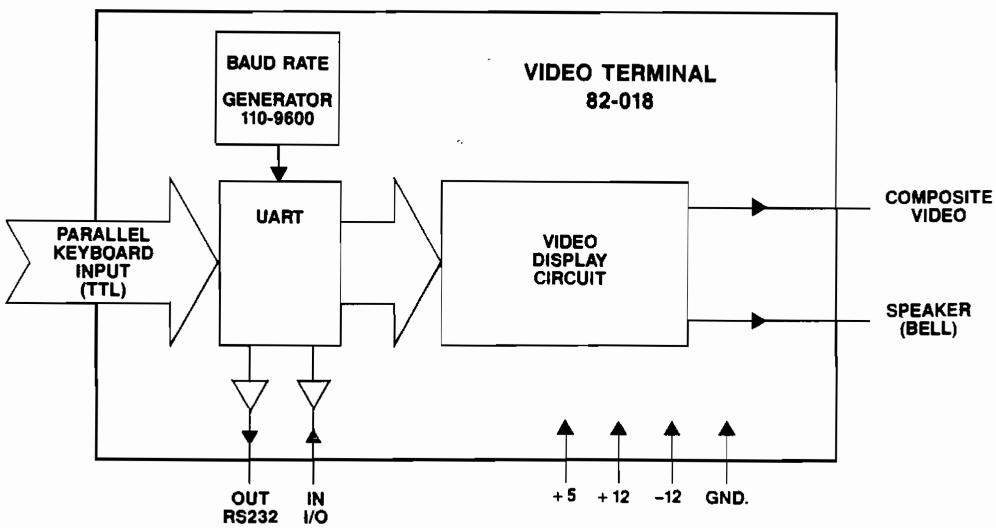
Complete source listing is included in the documentation. Both the character generator and the CRT program are in 2716 EPROMS to allow easy modification to your needs.

This board uses a 6502 Microprocessor and a 6545-1 CRT controller. The 6502 runs during the horz. and vert. blanking (45% of the time). The serial input port is interrupt driven. A 1500 character silo is used to store data until the 6502 can display it.



Features

- **6502 Microprocessor**
- **6545-1 CRT controller**
- **2716 EPROM char. gen.**
- **2716 EPROM program**
- **4K RAM (6116)**
- **2K EPROM 2716**
- **RS232 I/O for direct connection to computer or modem.**
- **80 columns x 25 line display**
- **Size 6.2" x 7.2"**
- **Output for speaker (bell)**
- **Power +5 700Ma.
+12 50Ma.
-12 50Ma.**



This board is available assembled and tested, or bare board with the two EPROMS and crystal.

Assembled and tested	#82-018A	$199.95
Bare board with EPROMS and crystal	#82-018B	$ 89.95

Both versions come with complete documentation.

JOHN BELL ENGINEERING, INC.

ALL PRODUCTS ARE AVAILABLE FROM **JOHN BELL ENGINEERING, INC.** • 1014 CENTER ST., SAN CARLOS, CA 94070
ADD SALES TAX IN CALIFORNIA • ADD 5% SHIPPING & HANDLING 3% FOR ORDERS OVER $100
10% OUTSIDE U.S.A.
ADD $1.50 FOR C.O.D.

MasterCard

SEND $1.00 FOR CATALOG

(415) 592-8411

WILL CALL HOURS: 9am - 4pm

VISA

#249

Circle No. 21

Centronics Printer Driver for Your Microcomputer

by Larry R. Hollibaugh

Get the most out of your Centronics-compatible printer with the MX-Driver assembly-language program described here.

MX-Driver
requires:

6502 microcomputer with one parallel port and a Centronics-compatible printer

The Epson MX-80 printer is touted in advertisements as the best-selling printer in the world. It is certainly an affordable dream-come-true for many computer hobbyists. The MX-80 has been improved twice since its introduction, making it an even greater value. These improvements come in the form of new ROM sets that plug into the logic board inside the printer. The Graftrax-80 option provides bit-map graphics at up to 120 dots-per-inch horizontally, by 72 dots-per-inch vertically. This option adds many features, including line spacing to n/216 of an inch, form length to 255 lines (original ROMs allow maximum form length of only 66 lines), slashed zeros, and an italics character set. Graftrax-Plus is the latest enhancement to both the MX-80 and the MX-100. Graftrax-Plus has most of the features of Graftrax-80 as well as underline, superscript, subscript, adjustable right margin, and more.

Here I explain how you connect your MX-80 printer to the Rockwell AIM 65. Included are the driver routines that provide access to the many features of your printer through the user device function of the AIM 65. With minor changes these routines can be used with printers other than Epson.

The MX-80 comes standard with a Centronics-compatible parallel interface. Details on this interface are provided in Appendix K of the original MX-80 user's manual, or Appendix 0 of the Graftrax-Plus manual. The interface consists of eight data lines, three handshaking lines, and several special printer, status, and control signals.

Connecting the MX-80 to the Rockwell AIM 65 computer through its parallel interface is straightforward and simple. Table 1 shows the connections to drive the MX-80 through Port B of AIM's User 6522 VIA (Versatile Interface Adaptor). Only the eight data lines and two of the handshake lines are required for basic communication. (Use of the other lines is left up to the reader's imagination.) A cable consisting of ten twisted pairs should be used between the computer and the printer. (Ribbon cable may be used if the cable is kept short.) At the printer end, using an Amphenol-type 57-30360 or equivalent connector, connect one line of each pair to each signal pin. Connect the other line of each pair to the associated return pin. (If using ribbon cable, alternate signal and ground lines.) At AIM's Application Connector, hook up the signal lines as shown in table 1 using a Vector-type R-644 or equivalent connector, and tie all the return lines to GND.

The STROBE line (pin 1) indicates to the printer that data is available, and the ACKNLG line (pin 10) signals when the printer is ready to accept more data. The 6522 VIA can be configured to handle this handshaking automatically. The software to handle the interface consists of two parts: a routine to initialize the port as outputs with proper handshaking, and a routine to transmit characters. These functions are provided by the routines CTINIT and CTOUT in listing 1.

The AIM firmware provides the ability to direct output to any of several output devices. The subroutine WHEREO ($E871) allows selection by

***Table 1:* Connections between the AIM 65's Application Connector and the MX-80's Centronics Connector.**

AIM 65 Appl. Conn.	AIM 65 Signal Name	Centronics Signal Pin	Centronics Return Pin	Centronics Signal Name
19	CB2	1	19	STROBE
9	PB0	2	20	DATA 1
10	PB1	3	21	DATA 2
11	PB2	4	22	DATA 3
12	PB3	5	23	DATA 4
13	PB4	6	24	DATA 5
16	PB5	7	25	DATA 6
17	PB6	8	26	DATA 7
15	PB7	9	27	DATA 8
18	CB1	10	28	ACKNLG
1	GND			

issuing the OUT= prompt. Answering this prompt with U (for User device) transfers control to the program pointed to by UOUT (at address $10A). This is a vector or hook that allows access to a user-defined output device. AIM's output subroutine OUTALL ($E9BC) now also transfers control through this hook.

The 6502's carry flag is clear for an initialization request from WHEREO, and set for character output from OUTALL. If you put the address of CTOT in listing 1 in the UOUT vector, you will have a basic Centronics-compatible parallel interface as the User Output Device. At CTOT, the carry flag is first tested to see if AIM wants to initialize the port or output a character. Initialization identifies itself, sets up the output port with full handshaking, and sends a couple of nulls to the printer to start the handshaking sequence. If the printer is off-line at this point, nothing more happens until it is put on-line. Output goes to CTOUT1 (the character from OUTALL is already on the stack). This routine waits if the printer is busy, off-line, or in an error condition, and sends the character when ready.

Listing 1 provides an alternate way to print text. When you put the address of MXOT in AIM's UOUT vector you get the MX-Driver, a sophisticated printer driver with skip-over-perf, a choice of six or eight lines per inch, the ability to define a left margin, and an easy way to select the various printer options. The initialization calls from WHEREO now go to MXINIT and output calls from OUTALL go to MXOUT.

MXINIT first calls the CTINIT subroutine discussed earlier, then requests lines per inch with the LPI = prompt. The four legal answers to this prompt are 6, 8, SPACE, and RETURN. Anything else makes the printer beep and the LPI= prompt is reissued. Answering 6 selects six lines per inch, 66 lines per page, and 62 lines of print per page (at which time a form feed is generated). Answering 8 selects eight lines per inch, 88 lines per page, and 82 lines of print per page. SPACE or RETURN here ends the initialization sequence. This provides an easy way to continue printing from where the last printing left off.

The next prompt, MARGIN=, allows you to specify a left margin width in decimal. Default is no margin. The last prompt is OPTS=. Here is where the real power of the MX-Driver

***Listing 1:* Basic Centronics interface and sophisticated printer driver for Epson MX-80 and MX-100 printers with Graftrax-Plus. Assembled to reside at the top of RAM on a 4K AIM 65.**

```
LOC   CODE       LINE

0000             ;    EPSON MX80/MX100 PRINTER DRIVER FOR THE AIM-65

0000             ;        BY LARRY R. HOLLIBAUGH

0000             ;    SUPPORTS TWO FORMS OF PRINTER COMMUNICATION:
0000             ;      1. BASIC CENTRONICS PARALLEL INTERFACE
0000             ;      2. SOPHISTICATED PRINTER DRIVER PROVIDING
0000             ;         INITIALIZATION OPTIONS, MARGINS, LINE
0000             ;         FEED SUPRESSION, AND SKIP-OVER-PERF

0000             ;    AIM SUBROUTINES AND MONITOR RAM

0000             OUTPUT =$E97A          ;ACCUM TO D/P
0000             REDOUT =$E973          ;READ KBD WITH ECHO
0000             RDRUB  =$E95F          ;READ KBD WITH DELETE ALLOWED
0000             CRLOW  =$EA13          ;CR LF TO D/P
0000             ADDIN  =$EAAE          ;GET FOUR DIGIT NUMBER
0000             PACK   =$EA84          ;PACK ASCII CHAR INTO HEX BYTE
0000             ADDR   =$A41C          ;'ADDIN' RESULT
0000             CURPO2 =$A415          ;DISPLAY CURSOR
0000             DIBUFF =$A438          ;DISPLAY BUFFER

0000             ;    AIM USER VIA ADDRESSES

0000             UDRB   =$A000          ;DATA REGISTER B
0000             UDDRB  =$A002          ;DATA DIRECTION REGISTER B
0000             UPCR   =$A00C          ;PERIPHERAL CONTROL REGISTER
0000             UIFR   =$A00D          ;INTERUPT FLAG REGISTER
0000             UIER   =$A00E          ;INTERUPT ENABLE REGISTER

0000             ;    EQUATES

0000             CR     =$D
0000             LF     =$A
0000             VTAB   =$B
0000             FFEED  =$C
0000             ESC    =$1B


0000                    *=$E00

0E00             ;    PROGRAM VARIABLES (MUST BE RAM)

0E00             LINCNT *=*+1           ;LINE COUNT
0E01             MRGFLG *=*+1           ;MARGIN FLAG (0=SET MARGIN)
0E02             LINMAX *=*+1           ;MAXIMUM LINES ON PAGE
0E03             MARGIN *=*+1           ;MARGIN WIDTH (IN BCD)
0E04             TEMP   *=*+1           ;TEMPORARY STORAGE


0E04             ;    ACCESS TO BASIC CENTRONICS PARALLEL INTERFACE
0E04             ;         THROUGH THE AIM USER OUTPUT VECTOR

0E05  B0 5A      CTOT   BCS CTOUT1      ;BRANCH ON CHARACTER OUTPUT

0E07  A0 00             LDY #CNTMSG-LITS
0E09  20 92 0F          JSR PMSG        ;IDENTIFY SELF
0E0C  EE 01 0E          INC MRGFLG      ;DO NEXT FFEED THRU MXOUT

0E0F             ;    CTINIT : INITIALIZE PARALLEL PORT

0E0F  A9 FF      CTINIT LDA #$FF
0E11  8D 02 A0          STA UDDRB       ;ALL BITS OUTPUT
0E14  A9 18             LDA #$18
0E16  8D 0E A0          STA UIER        ;DISABLE INTERUPTS (JUST IN CASE)
0E19  AD 0C A0          LDA UPCR
0E1C  29 0F             AND #$F         ;SET CB1 TO NEGATIVE EDGE DETECT
0E1E  09 A0             ORA #$A0        ;SET CB2 TO AUTO PULSE MODE
0E20  8D 0C A0          STA UPCR

0E23  A9 00             LDA #0          ;SEND A NULL TO GET THINGS STARTED
0E25  8D 00 A0          STA UDRB
0E28  4C 60 0E          JMP CTOUT       ;SEND ANOTHER, TO BE SURE


0E2B             ;       ENTRY TO USE THE MX-DRIVER THROUGH THE
0E2B             ;            AIM USER OUTPUT VECTOR

0E2B  B0 40      MXOT   BCS MXOUT       ;BRANCH ON CHARACTER OUTPUT

0E2D             ;      MXINIT : INITIALIZE THE PRINTER FOR OUTPUT

0E2D  20 0F 0E   MXINIT JSR CTINIT      ;INITIALIZE CENTRONICS PORT
0E30  A0 0D      GETLPI LDY #LPIMSG-LITS
0E32  20 92 0F          JSR PMSG        ;REQUEST LINES PER INCH
0E35  20 73 E9          JSR REDOUT      ;GET REPLY
0E38  C9 38             CMP #'8'        ;8 LINES PER INCH?
0E3A  F0 11             BEQ SETPRM
0E3C  C9 36             CMP #'6'        ;6 LINES PER INCH?
0E3E  F0 0D             BEQ SETPRM
0E40  C9 20             CMP #' '        ;SPACE MEANS USE OLD PARAMETERS
0E42  F0 17             BEQ OLDPRM
0E44  C9 0D             CMP #CR         ;RETURN ALSO
0E46  F0 13             BEQ OLDPRM
0E48  20 5E 0E          JSR BELL        ;MUST BE ERROR
0E4B  D0 E3             BNE GETLPI      ;ASK AGAIN

0E4D  48         SETPRM PHA             ;SAVE LPI
0E4E  20 D5 0E          JSR GETMRG      ;GET MARGIN WIDTH
0E51  20 FF 0E          JSR GETOPT      ;GET OPTIONS
0E54  68                PLA
0E55  20 2C 0F          JSR SETMX       ;RESET PRINTER AND SET LPI
0E58  20 4F 0F          JSR SNDOPT      ;SEND REQUESTED OPTIONS
0E5B  4C 13 EA   OLDPRM JMP CRLOW


0E5E             ;    BELL : RING BELL ON PRINTER

0E5E  A9 07      BELL   LDA #7          ;ASCII BELL CHAR

0E60             ;    CTOUT : SEND CHARACTER TO CENTRONICS PORT

0E60  48         CTOUT  PHA
0E61  AD 0D A0   CTOUT1 LDA UIFR        ;GET VIA STATUS
0E64  29 10             AND #$10        ;IS PRINTER READY?
0E66  F0 F9             BEQ CTOUT1      ;NO, WAIT
0E68  68                PLA
0E69  8D 00 A0          STA UDRB        ;YES, SEND CHARACTER
0E6C  60                RTS

0E6D             ;    MXOUT : SEND CHAR, WITH MARGINS AND SKIP-OVER-PERF
```

Listing 1 *(continued)*

```
LOC   CODE       LINE
0E6D  68         MXOUT   PLA
0E6E  C9 FF              CMP #$FF        ;IGNORE AIM'S NULL CODES
0E70  F0 56              BEQ MXDUN

0E72  48         MXOUT1  PHA
0E73  29 7F              AND #$7F        ;STRIP MSB
0E75  C9 0A              CMP #LF         ;IGNORE LINE FEEDS
0E77  F0 4E              BEQ MXRET
0E79  C9 0C              CMP #FFEED      ;FORM FEED?
0E7B  D0 0C              BNE MXOUT2      ;NOPE
0E7D  AD 00 0E           LDA LINCNT      ;ALREADY AT TOP OF FORM?
0E80  D0 22              BNE MXOUT3      ;NOPE
0E82  AD 01 0E           LDA MRGFLG      ;FIRST CHAR IN LINE?
0E85  D0 1D              BNE MXOUT3      ;NO, DO IT
0E87  F0 45              BEQ SMF         ;YES, SUPRESS EXTRAS

0E89  AD 01 0E   MXOUT2  LDA MRGFLG      ;TIME TO DO MARGIN?
0E8C  D0 16              BNE MXOUT3      ;NOT YET
0E8E  EE 01 0E           INC MRGFLG      ;YES, CLEAR FLAG
0E91  AD 03 0E           LDA MARGIN
0E94  F8         MRGLUP  SED
0E95  38                 SEC

0E96  E9 01              SBC #1          ;DECREMENT IN DECIMAL
0E98  D8                 CLD
0E99  30 09              BMI MXOUT3      ;DONE IF PAST ZERO
0E9B  48                 PHA
0E9C  A9 20              LDA #' '
0E9E  20 60 0E           JSR CTOUT       ;PRINT A SPACE
0EA1  68                 PLA
0EA2  10 F0              BPL MRGLUP      ;AND LOOP

0EA4  68         MXOUT3  PLA
0EA5  48                 PHA
0EA6  20 60 0E           JSR CTOUT       ;SO, PRINT IT
0EA9  29 7F              AND #$7F        ;STRIP MSB
0EAB  C9 0C              CMP #FFEED      ;FORM FEED?
0EAD  F0 1A              BEQ NEWPAG      ;YES, RESET TOP OF PAGE
0EAF  C9 0D              CMP #CR         ;RETURN?
0EB1  F0 04              BEQ NEWLIN      ;YES
0EB3  C9 0B              CMP #VTAB       ;VERTICAL TAB?
0EB5  D0 10              BNE MXRET       ;NO, WE'RE DONE
0EB7  EE 00 0E   NEWLIN  INC LINCNT      ;COUNT LINE
0EBA  AD 00 0E           LDA LINCNT
0EBD  CD 02 0E           CMP LINMAX      ;SKIP OVER PERF YET?
0EC0  90 0C              BCC SMF         ;NOT YET
0EC2  A9 0C              LDA #FFEED      ;YES, DO FORM FEED
0EC4  20 72 0E           JSR MXOUT1
0EC7  68         MXRET   PLA
0EC8  60         MXDUN   RTS

0EC9  A9 00      NEWPAG  LDA #0          ;ZERO LINE COUNT
0ECB  8D 00 0E           STA LINCNT
0ECE  A9 00      SMF     LDA #0          ;SET MARGIN FLAG
0ED0  8D 01 0E           STA MRGFLG
0ED3  68                 PLA
0ED4  60                 RTS

0ED5                     ;   GETMRG : GET MARGIN WIDTH

0ED5  AD 1C A4   GETMRG  LDA ADDR        ;SAVE ADDR
0ED8  48                 PHA
0ED9  AD 1D A4           LDA ADDR+1
0EDC  48                 PHA
0EDD  20 13 EA   GTMRG1  JSR CRLOW
0EE0  A0 13              LDY #MRGMSG-LITS
0EE2  20 92 0F           JSR PMSG        ;REQUEST MARGIN
0EE5  20 AE EA           JSR ADDIN       ;GET IT
0EE8  90 06              BCC MRGOK
0EEA  20 5E 0E           JSR BELL        ;SIGNAL ERROR
0EED  4C DD 0E           JMP GTMRG1      ;AND ASK AGAIN

0EF0  AD 1C A4   MRGOK   LDA ADDR        ;USE ONLY LAST 2 NUMBERS ENTERED
0EF3  8D 03 0E           STA MARGIN
0EF6  68                 PLA             ;RESTORE ADDR
0EF7  8D 1D A4           STA ADDR+1
0EFA  68                 PLA
0EFB  8D 1C A4           STA ADDR
0EFE  60                 RTS

0EFF                     ;   GETOPT : GET OPTIONAL PRINTER MODES

0EFF  A0 1B      GETOPT  LDY #OPTMSG-LITS
0F01  20 92 0F           JSR PMSG        ;REQUEST OPTIONS
0F04  AD 15 A4           LDA CURPO2
0F07  8D 04 0E           STA TEMP        ;SAVE STARTING BUFFER POS
0F0A  A0 00              LDY #0
0F0C  20 5F E9   OPT1    JSR RDRUB       ;INPUT CHARACTERS
0F0F  C9 20              CMP #' '        ;UNTIL SPACE
0F11  F0 07              BEQ OPTIN
0F13  C9 0D              CMP #CR         ;OR RETURN
0F15  F0 03              BEQ OPTIN
0F17  C8                 INY             ;NEXT CHARACTER
0F18  10 F2              BPL OPT1

0F1A  98         OPTIN   TYA

0F1B  18                 CLC
0F1C  6D 04 0E           ADC TEMP        ;CALC TRUE BUFFER INDEX
0F1F  C9 3C              CMP #60         ;PAST END OF DIBUFF?
0F21  90 02              BCC OPTOK
0F23  A9 3B              LDA #59         ;STOP HERE
0F25  A8         OPTOK   TAY
0F26  A9 00              LDA #0
0F28  99 38 A4           STA DIBUFF,Y    ;MARK END
0F2B  60                 RTS

0F2C                     ;   SETMX : RESET MX80 AND SET LINES PER INCH

0F2C  48         SETMX   PHA             ;SAVE REQUESTED LPI
0F2D  A9 0C              LDA #FFEED
0F2F  20 72 0E           JSR MXOUT1      ;GO TO TOP OF FORM
0F32  A2 00              LDX #RESET-INITS
0F34  20 3E 0F           JSR MXILUP      ;NOW RESET PRINTER
0F37  68                 PLA
0F38  C9 38              CMP #'8'        ;8 LPI REQUESTED?
0F3A  D0 8C              BNE MXDUN       ;NO
0F3C  A2 09              LDX #LPI8-INITS ;SET 8 LPI, 82 LPP

0F3E                     ;   MXILUP : SEND INIT PARMS TO PRINTER VIA X

0F3E  BD C0 0F   MXILUP  LDA INITS,X     ;NEXT PARM
0F41  30 06              BMI SETLPP      ;QUIT ON MINUS
0F43  20 60 0E           JSR CTOUT       ;SEND TO PRINTER
0F46  E8                 INX
0F47  D0 F5              BNE MXILUP      ;BRANCH ALWAYS
0F49  29 7F      SETLPP  AND #$7F        ;STRIP MSB
0F4B  8D 02 0E           STA LINMAX      ;MAX LINES OF PRINT PER PAGE
0F4E  60                 RTS
```

comes into play. You can select any of the escape sequences and control codes recognized by the printer, and change the number of lines of print per page. To select an escape sequence, enter the ASCII character that follows the escape. (Refer to the printer manual for the proper codes.) For example, the combination of emphasized and double-strike print modes produces a high quality print. On the Epson ESC,E enables emphasized and ESC,G enables double strike, so typing EG (followed by SPACE or RETURN) sets these modes. Absolute hex numbers can also be sent by typing a dollar sign ($) followed by two hex digits. In this way, control codes and complex escape sequences can be sent. With Graftrax-Plus, you can print at 66 characters per eight-inch line by enabling double-width and compressed characters at the same time. To do this, type W$01$0F. W sends ESC, W (double-width mode control), $01 turns the mode on, and $0F is CTRL-O (enable compressed). Change the lines of print per page by preceding two hex digits with a single quote mark ('). This number is not sent to the printer but is used by the MX-Driver for forms control. To set the MX-80 to eight dots per line feed, 99 lines per page, with 93 lines of print per page, enter A$08C$63'5D for Graftrax-80, or A$08'5D for Graftrax-Plus. (With Graftrax-Plus, the form length has already been set to 11 inches.) Be careful when answering this prompt, as it does no error checking. Use the DELETE key to edit, and type SPACE or RETURN when correct.

Now the MX-Driver sends a form feed and resets the printer. It sets up for eight lines per inch if selected and sends the requested options.

As characters are sent out by OUTALL, they go to MXOUT. This routine ignores the nulls ($FF) and line feeds sent out by AIM's CRLF routine when in TTY mode. Be sure the MX-80 is set up to do a line feed automatically with carriage return by setting switch 2-3 ON. (See the original MX-80 user's manual, Appendix C, or the Graftrax-Plus manual, Appendix E.) This way the printer operates properly whether AIM is in KB or TTY mode, since the mode affects whether or not OUTALL sends line feeds.

MXOUT will not send two successive form feeds. This saves you from shoving out a whole blank page if the last job on the printer left the paper at the top of a form. Margins are now inserted if indicated by the last character having been a carriage return, vertical tab, or form feed, and the character to be printed is sent *via* the CTOUT subroutine. Finally, MXOUT checks for form feed, carriage return, and vertical tab characters for special handling. Form feed zeros out the line count and flags for a margin. Carriage return or vertical tab advances the line count, generates a form feed if the maximum number of lines have been printed, and flags for a margin.

Now you have two ways to send characters to your printer. If the program sending the characters to the Active Output Device is doing all of its own formatting and control, or sending graphics or other non-ASCII, you can use the bare-bones Centronics interface. Every character sent out this way goes direcly to the printer.

To print ASCII text that knows nothing of the special requirements and capabilities of the printer, you can use the MX-Driver to handle the basic formatting. Unfortunately, this involves modifying the UOUT vector every time

Listing 2 *(continued)*

```
LOC   CODE      LINE
0F4F            ;    SNDOPT : SEND REQUESTED OPTIONS TO PRINTER

0F4F  AC 04 0E  SNDOPT LDY TEMP        ;INDEX INTO DISPLAY BUFFER
0F52  B9 38 A4  NXTOPT LDA DIBUFF,Y    ;GET NEXT OPTION
0F55  F0 3A            BEQ OPTRET      ;ZERO MARKS END
0F57  C9 24            CMP #'$'        ;HEX NUMBER?
0F59  F0 04            BEQ HEXNUM      ;YES
0F5B  C9 27            CMP #$27        ;SINGLE QUOTE FOR CHG LINMAX?
0F5D  D0 25            BNE ESCSEQ

0F5F  8D 04 0E  HEXNUM STA TEMP        ;SAVE BYTE USE MARKER
0F62  A2 02            LDX #2          ;2 CHARACTERS TO PACK
0F64  C8        PKBYT  INY
0F65  B9 38 A4         LDA DIBUFF,Y    ;NEXT CHARACTER
0F68  F0 27            BEQ OPTRET      ;ZERO MARKS END
0F6A  20 84 EA         JSR PACK        ;PACK IT IN
0F6D  B0 15            BCS ESCSEQ      ;NOT HEX
0F6F  CA               DEX             ;COUNT IT
0F70  D0 F2            BNE PKBYT       ;AND LOOP
0F72  AE 04 0E         LDX TEMP
0F75  E0 24            CPX #'$'        ;TO PRINTER?
0F77  F0 05            BEQ SNDBYT
0F79  8D 02 0E         STA LINMAX      ;NEW MAX LINES PER PAGE
0F7C  D0 10            BNE ESC1        ;BRANCH ALWAYS

0F7E  20 60 0E  SNDBYT JSR CTOUT       ;SEND TO PRINTER
0F81  4C 8E 0F         JMP ESC1

0F84  48        ESCSEQ PHA             ;SEND AS ESCAPE SEQUENCE
0F85  A9 1B            LDA #ESC
0F87  20 60 0E         JSR CTOUT       ;SEND ESCAPE FIRST
0F8A  68               PLA
0F8B  20 60 0E         JSR CTOUT       ;AND NOW CHARACTER
0F8E  C8        ESC1   INY
0F8F  10 C1            BPL NXTOPT      ;LOOP
0F91  60        OPTRET RTS

0F92            ;    PMSG : DISPLAY MESSAGE FROM LITERAL TABLE VIA Y

0F92  B9 9E 0F  PMSG   LDA LITS,Y      ;GET A CHAR
0F95  F0 06            BEQ PDUN        ;QUIT ON ZERO
0F97  20 7A E9         JSR OUTPUT      ;SEND IT
0F9A  C8               INY
0F9B  D0 F5            BNE PMSG        ;NEXT CHAR
0F9D  60        PDUN   RTS

0F9E            ;    LITERAL TABLE

0F9E            LITS   =*
0F9E  20 43     CNTMSG .BYT ' CENTRONICS ',0
0FA0  00
0FAB  20 4C     LPIMSG .BYT ' LPI=',0
0FAD  00
0FB1  20 4D     MRGMSG .BYT ' MARGIN',0
0FB3  00
0FB9  20 4F     OPTMSG .BYT ' OPTS=',0
0FBB  00
```

Listing 1 *(continued)*

```
LOC   CODE  LINE
0FBC         ;    TABLE OF INITIALIZATION PARMS FOR GRAFTRAX-PLUS

0FC0        INITS  =*
0FC0  1B    RESET  .BYT ESC,'@'       ;RESET PRINTER
0FC1  40
0FC2  1B           .BYT ESC,'O'       ;RESET SOP (WE DO OUR OWN)
0FC3  4F
0FC4  1B           .BYT ESC,'C',0,11;FORM LENGTH 11 INCHES
0FC5  43
0FC6  00
0FC7  0B
0FC8  BE           .BYT 62+128        ;62 ACTIVE LINES PER PAGE
0FC9  1B    LPI8   .BYT ESC,'0'       ;8 LINES PER INCH
0FCA  30
0FCB  D2           .BYT 82+128        ;82 ACTIVE LINES PER PAGE

0FCC               .END
```

Listing 2: **Alternate printer initialization table for Epson MX-80 with Graftrax-80. Replaces table at the end of listing 1.**

```
0FBC         ;    TABLE OF INITIALIZATION PARMS FOR GRAFTRAX-80

0FC0        INITS  =*
0FC0  1B    RESET  .BYT ESC,'='       ;CLEAR M.S.B. FUNCTION
0FC1  3D
0FC2  1B           .BYT ESC,'5'       ;ENABLE STANDARD CHAR SET
0FC3  35
0FC4  1B           .BYT ESC,'F'       ;CANCEL EMPHASIZED MODE
0FC5  46
0FC6  1B           .BYT ESC,'H'       ;CANCEL DOUBLE STRIKE MODE
0FC7  48
0FC8  12           .BYT $12           ;CANCEL COMPRESSED CHAR MODE
0FC9  1B           .BYT ESC,'2'       ;6 LINES PER INCH
0FCA  32
0FCB  1B           .BYT ESC,'C',66 ;66 LINES PER PAGE
0FCC  43
0FCD  42
0FCE  BE           .BYT 62+128        ;62 ACTIVE LINES PER PAGE
0FCF  1B    LPI8   .BYT ESC,'0'       ;8 LINES PER INCH
0FD0  30
0FD1  1B           .BYT ESC,'C',88 ;88 LINES PER PAGE
0FD2  43
0FD3  58
0FD4  D2           .BYT 82+128        ;82 ACTIVE LINES PER PAGE
```

you want a different interface. If you have a disk or a modem, etc., there is even more switching and potential for mistakes. In the April 1982 issue of MICRO (47:06) there is an article by Joel Swank entitled "AIM User Device Arbiter" describing selection of multiple input and output devices. The UDA gives a DEVICE = prompt in reponse to answering the IN = or OUT = prompt with U. Joel's article explains how to access the two interfaces presented here, and any others you may have on your system, by putting a single letter code and a two-byte address for each device handler in an output device table. I call the Centronics interface C and the MX-Driver P.

As presented in listing 1, the MX-Driver works on MX-80's and MX-100's with Graftrax-Plus. For anything else, you need to change the table of initialization parameters at the end of the program. This table contains two strings, labeled RESET and LPI8. Each string ends with a byte that has the most significant bit set. The other seven bits of this byte are used as the default lines of print per page by the MX-Driver. The other characters at RESET are sent to the printer to reset it to the desired defaults (for six lines per inch), and those at LPI8 are sent (if requested) to set up the eight lines-per-inch option. At RESET in listing 1 the printer is reset to power-on defaults, skip-over-perf is disabled (since the MX-Driver handles it), and form length is set to 11 inches.

Listing 2 shows the alternate table for Graftrax-80. Graftrax-80 has a bug in the "reset printer" escape sequence (ESC,@) that causes the top of form to creep. If you don't know whether or not your MX-80 has this bug, turn it off and back on. If the paper moves, it's got the bug! At RESET in listing 2 the printer is reset by turning off the various modes it could be in — M.S.B., italics, emphasized, double strike, and compressed character modes—and setting up the default of six lines per inch, 66 lines per page. For those without an assembler, the only references to the initilization table are in the SETMX subroutine. The offsets into the table need to be adjusted when the table is changed. By changing this table, the MX-Driver can be used with many printers other than Epson.

Larry R. Hollibaugh has been a computer hobbyist for the last five years, learning by building up the AIM 65 into a dedicated controller development system complete with video, disk, and printer. He is employed as a field service technician repairing coin-operated amusement devices. You may contact him at 1206 S.E. Harney St., Portland, OR 97202.

MICRO™

Disk ID for Printed OSI Directories

by Robert A. Paul

Utilize the open area of OS65D3 directory sector #1 to allow inclusion of diskette ID on printed directories.

When you use large numbers of disks, it soon becomes obvious that you need hard copies of the directory for each disk on file. Furthermore, you need to correlate the printed sheets directly with the specific disk to which each refers.

While examining several of the utilities for my OSI Superboard-MF, I noticed that OSI has taken utilities that were probably written for 8" disks and only slightly modified them for 5¼"-disk systems. For instance, the directory-oriented utilities contained on my 5¼" disks all include two calls to track 12, sectors 1 and 2. For 5¼" disks this is not necessary:

Track Numbers	Utility Name
0 - 12	OS65D
13	track 0 r/w
14	BEXEC*

Total number of tracks used: 15 out of a maximum of 40. Total number of entries used by OSI: 2 of 32 available.

This leaves 25 additional entries required in the directory even if all the rest of the programs on the disk are only one track long. Therefore, a total of 27 entries are needed. If the contents of the directory buffer are examined after a call to track 12,1 has been made, you can see that there are spaces for 32 8-character zones including two characters per name, which set the first and last track of the program. If you stay with the original 6-character program names, only one call to track 12,1 will give you more than enough spaces for all the programs that you can access by track name. Deleting the second call from the directory-oriented utilities will reduce the wear and tear on the disk, and will slightly speed up the printing process. However, the main insight here is that you now have a location where disk data such as ID name or number can be stored, and you have automatic access to it when doing a printed directory listing without making "calls" to other tracks or sectors. The following lines will need to be included in any program to access and print this information:

```
DISK !"CA 2E79=12,1"
D$ = ""
FOR J = 12143 TO 12152
D$ = D$ + CHR$(PEEK(J))
NEXT J
——
——
PRINT #DV,"- DIRECTORY -";D$
——
——
——
FOR I = PN TO PN+240 STEP8
```

If you are using a BEXEC* that supports up to 14-character directory names, the same idea should work, but you will have to make that second call to track 12,2 after picking out the stored ID word at the end of the buffer filled by the first call.

Sample Directory

OS-65D Ver-3.0 6-2-82
- Directory - OS65D #17

File Name	Track Range
OS65D3	0 - 13
BEXEC*	14 - 14
CHANGE	15 - 15
CREATE	16 - 16
DELETE	17 - 17
DIR	21 - 21
DIRSRT	22 - 22
FILIST	23 - 23
RENAME	24 - 24
TRACE	25 - 25
ALCO	34 - 35
LG15	36 - 38
SORT	39 - 39

18 entries free of 32

The ID information must be put on the directory track as follows:

```
OK
EXIT
01 TRACK
A*CA 2E79=12,1

A*EM

EM  V2.0

:@2F6F

2F6F  23 20
2F70  23 4F     O
2F71  23 53     S
2F72  23 36     6
2F73  23 35     5
2F74  23 44     D
2F75  23 20
2F76  23 (LF)   #
2F77  23 31     1
2F78  23 37     7
2F79  23 (RET)
:EX

A*SA 12,1=2E79/1
```

You may want to change the protection of the track 0 read/write code in track 13 by placing a 13 @ 2E80 before going on to the changes shown for 2F6F and on. The next time you run a directory it will be included with OS65D3.

To modify some of the utilities involving access of the directory track information change line 10030 to show 240 instead of 248 in the DELETE, ZERO, and RENAME utilities. Also, line 11040 in DIRSRT needs to reflect this same change, while line 10030 in CREATE should contain 246 instead of 254. These changes will prevent any interference with the location of your new ID. Finally, you should eliminate line 10130 in DIR and DIRSRT, or modify the 64 in that line to a 32.

You may contact the author at 4406 Bridle Rd., Bartlesville, OK 74003.

MICRO

Circle No. 25

Circle No. 26

Circle No. 27

Attention

OSI Users!

YOU'VE BEEN WAITING FOR IT . . .
HERE IT IS!!!

A Brand New Book...The Last Word in OSI Information!

Improvements for your Ohio Scientific Computer including Machine Language enhancements and BASIC Aids. TRACE BASIC programs, DEBUG machine language programs and improve your OS65-D operating system. Hardware modifications for enhanced/reversed video, programs for control code and upper/lower case entry, and a What's Where in the OSI C1/C2 are just some of the useful information presented. Improve your own BASIC programs with PRINT AT, DELETE, AUTO-NUMBER and FUNCTION INPUT. No OSI owner can afford to be without this book.

25 Programs — 176 Pages — Only $19.95

USE COUPON TO ORDER TODAY DIRECTLY FROM MICRO

Mail To: **MICRO, 34 Chelmsford St., P.O. Box 6502, Chelmsford, MA 01824**

Yes! Send me ________ MICRO on the OSI @ $19.95 (plus $ 2 s/h)

Name ____________________

Address ____________________

City ____________ State ________ Zip ____________

Card # ____________________ Expires ____________

☐ Check Enclosed
☐ VISA
☐ MasterCard

Or Call Toll Free:
1-800-345-8112
(In PA 1-800-662-2444)

MA residents add 5% sales tax. Allow 6-8 weeks for delivery.

APPLE PRINT-USING Routine

by Celestino R. Monclova

A machine-language routine to simulate PRINT-USING from Applesoft BASIC.

PRINT-USING
requires:
Apple II with Applesoft

When called from BASIC, this machine-language routine has the capability of merging two string variables (a mask string and a data string) with the following options:

1. Fixed decimal points and commas
2. Fixed dollar sign and text
3. Asterisk or dollar sign fill
4. Floating dollar sign

Options 1, 2, and 3 are controlled by the way you set up your mask string and may be used in any combination with option 4, which is handled by appending your "float" character to the data string — all in BASIC.

On the mainframe computer, the floating dollar sign and asterisk are handled by setting hardware flags. By eliminating the need for flags in this machine-language code, I feel I have improved upon the mainframe version by allowing any and all ASCII characters to float and/or fill.

Included with the assembly listing is a short program that displays a set array of number-strings and the result when they are combined with various user masks, which are keyed into this routine. The first mask, "TOTAL AMOUNT IS $, , .00", demonstrates the prime use of this routine and options 1 and 2 above. The data strings show that the unused commas will be suppressed. The mask is not limited to numeric characters only.

The second mask, "TOTAL TO DATE $$,$$$,$$$.0000", demonstrates dollar sign or any character fill. The fill characters still suppress the unneeded commas, and the zeros after the decimal point are not limited to two. In fact, you don't even need a decimal point, as shown in the third and fourth masks, "TOTALS " and "TOTALS $$$$$$$$$$$$". Nor do you need zeros after a decimal point (mask five, " ").

The last mask, "TOTALS.......,...,....00", shows flexibility by using decimal point fill.

The BASIC interface to this routine is the technique that assigns X$ = " " at the beginning of your BASIC program. Later, when you are ready to edit your data, you must first assign your mask X$ = "totals , , .00". The mask *must* be reassigned each time you edit. This is because the mask is modified and not your data string.

Next your data string is defined as any string variable available to BASIC. If you have a numeric variable, then use the instruction "Y$ = STR$ (X)" where X is the variable you wish to edit. No check is made on the numeric quality of this data string, so an instruction like "Y$ = "$" + STR$(X)" will give you the floating dollar sign.

After X$ has been assigned as the first variable at the beginning of your BASIC program and the machine-language routine has been loaded, the actual usage could be:

```
100 X$ = "***,***.00" : CALL 27904Y$
: PRINT X$
```

The lack of space, comma, or colon between the CALL and your data name is important.

This routine, which converts numeric strings into English, makes use of the presently unused input buffer page. If you want to defer printing after the CALL instruction to do INPUT, or save the newly edited string for reuse, then you must save it from its volatile position in the input buffer by assigning it to another variable: Y$ = MID$(X$,1). Listing 3 is a short program demonstration. The BASIC listing depends upon the PRINT-USING routine to align the numbers into columns as previously demonstrated, and can be considered an extended application for the machine-language routine.

Lines 600 and 625 set up X$ and read in the English words from the data strings at lines 1000 and 1002. Line 745 sets up the conversion mask that is interpreted by X and Y in the following manner:

```
X$ = " 000000.00"
X  =   0     1
Y  =   012012
```

Thus, the X,Y FOR NEXT loops interpret the data string from the A$ and B$ arrays. By extending the mask and including a C$ string, conversion can be easily expanded to many times the national debt.

The machine-language routine starting at location $6D00 is totally relocatable to any other unused page in memory. Lines 2 through 12 locate the mask X$, move its length to $1A, and start adding to locations $8 and $9. Line 5 also stores the length in location $1C, if there is no decimal point found.

Moving X$ to page $200 resolves several problems. The first problem is how to keep track of all the necessary pointers. Locating the beginning of the mask at $200 eliminates that pointer. The second problem, as some of the examples demonstrate, is how to keep the data string, when it overlaps the mask

(continued on pg. 41)

Listing 1

```
6DØØ                   ORG $6DØØ
6DØØ                   OBJ $8ØØ
6DØØ AØØ2              LDY #2
6DØ2 841B              STY $1B
6DØ4 B169              LDA ($69),Y     *FIND X$
6DØ6 851C              STA $1C         *SAVE LENGTH
6DØ8 851A              STA $1A
6DØA C8                INY
6DØB B169              LDA ($69),Y
6DØD 85Ø8              STA $8          *AND ADDRESS (H)
6DØF C8                INY
6D1Ø B169              LDA ($69),Y
6D12 85Ø9              STA $9          *AND ADDRESS (L)
6D14 AØØØ              LDY #Ø
6D16           *FIND X$ "."WHILE MOVING TO $2ØØ
6D16 B1Ø8      PERIOD  LDA ($8),Y
6D18 C92E              CMP #$2E
6D1A DØØ2              BNE MOVE
6D1C 841C              STY $1C
6D1E 99ØØØ2    MOVE    STA $2ØØ,Y
6D21 C8                INY
6D22 C41A              CPY $1A
6D24 9ØFØ              BCC PERIOD
6D26           *MOVED WITH PERIOD AND POINTER AT $1A
6D26           *GET DATA AND FIND "."
6D26 2ØE3DF            JSR $DFE3       *FIND DATA ADDRESS
6D29 AØØØ              LDY #Ø
6D2B B183              LDA ($83),Y
6D2D 8587              STA $87
6D2F C8                INY
6D3Ø B183              LDA ($83),Y
6D32 8585              STA $85
6D34 C8                INY
6D35 B183              LDA ($83),Y
6D37 8586              STA $86
6D39 AØØØ              LDY #Ø
6D3B B185      ALIGN   LDA ($85),Y
6D3D C92E              CMP #$2E        *"."
6D3F FØØ7              BEQ FOUND
6D41 C8                INY
6D42 C487              CPY $87
6D44 3ØF5              BMI ALIGN
6D46 A487              LDY $87
6D48 8419      FOUND   STY $19
6D4A A51C              LDA $1C
6D4C 38                SEC
6D4D E519              SBC $19
6D4F 851A              STA $1A
6D51           *OFFSET FROM 2ØØ FOR USER DATA
6D51 A487              LDY $87
6D53 88        COMMA   DEY
6D54 B11A              LDA ($1A),Y
6D56 C92C              CMP #$2C        *","
6D58 DØØ2              BNE MOVZE
6D5A C61A              DEC $1A
6D5C B185      MOVZE   LDA ($85),Y
6D5E 911A              STA ($1A),Y
6D6Ø 98                TYA
6D61 DØFØ              BNE COMMA
6D63           *CLEAR REMAINING COMMAS
6D63 A41A              LDY $1A
6D65 A9ØØ              LDA #Ø
6D67 851A              STA $1A
6D69 B11A      SKIP    LDA ($1A),Y
6D6B C92C              CMP #$2C        *","
6D6D FØØ6              BEQ CLEAR
6D6F C94Ø              CMP #$4Ø
6D71 1ØØB              BPL END
6D73 3ØØ6              BMI ENDØ1
6D75 88        CLEAR   DEY
6D76 B11A              LDA ($1A),Y
6D78 C8                INY
6D79 911A              STA ($1A),Y
6D7B 88        ENDØ1   DEY
6D7C DØEB              BNE SKIP
6D7E AØØ3      END     LDY #3
6D8Ø A9ØØ              LDA #Ø
6D82 9169              STA ($69),Y
6D84 C8                INY
6D85 A9Ø2              LDA #2
6D87 9169              STA ($69),Y
6D89 6Ø                RTS
```

(continued from pg. 39)

on one end or the other, from destroying adjacent strings in variable memory (more pointers?). The third problem involves scanning the mask string to find the decimal point. Moving and isolating it does not cost much processing time, and the space is available anyway.

Starting at line 16, the subroutine called PERIOD moves X$ to $200 and, at the same time, records the last period at location $1C. Lines 27 through 36 locate the data string and save its length at $87 and its address at $85,$86.

Subroutine ALIGN at line 38 searches forward on the data string for its decimal point. Finding none at line 44, it defaults to the right end.

Now that you have located the data decimal point at FOUND, compute into $1A the left starting position of the data string over the mask string. However, if you start moving the data string at the $1B left end, you will miss out on a lot of other goodies. Location $1A-$1B (set up at line 3) is the pointer to the left-end position of where the data string should start. Loading the Y register with the length at $87 gives you the first data character at the right end. There is plenty of room in memory if this position should overlap the mask string.

Subroutine COMMA starts the move by first checking the receiving position of the mask for a comma. Finding none, it jumps to MOVZE and moves the rightmost data bytes to the mask. If it finds a comma in the mask area before finishing the data string, this loop decrements the memory pointer at $1A to skip it and stores the data with MOVZE anyway at the next mask location.

Clean-up involves checking the remaining length of the mask for commas — namely, from whatever length is left in $1A to zero. If it is a comma, then CLEAR uses as the fill character whatever character is in the mask that preceeded the comma.

Line 67 checks for any characters in the mask. This ensures that commas in the mask are not cleared (i.e., a mask of X$ = "TOTALS,   ,   .00"). This keeps the comma after the word "TOTALS" from being edited out.

Finally, END forces the start address of the new X$ to $200. The length remains the same and you return to the BASIC program.

Listing 2

```
10 X$ = "": REM  FIRST VARIABLE
50  PRINT  CHR$ (4)"BLOAD PRINT U
     SING"
100  FOR X = 1 TO 10: READ Y$(X):
      NEXT
150  INPUT Y$: REM  GET DEMO MASK
200  FOR X = 1 TO 10:X$ = Y$: CALL
     27904Y$(X)
250  PRINT X$"  "Y$(X): NEXT
300  GOTO 150
350  DATA  12345678,123456.8,1234
     5.78,1234.678,123.5678,12.45
     678,.01234567
400  DATA .00012345,$12345.78,$12
     3.5678
```

Listing 3

```
600 X$ = "": DIM A$(20)
610  PRINT  CHR$ (4)"BLOAD PRINT
     USING"
625  FOR X = 0 TO 19: READ A$(X):
      NEXT : FOR X = 0 TO 9: READ
     B$(X): NEXT
700  INPUT Z$
730 X$ = "$,$$$,$$0.00": CALL 279
     04Z$: PRINT X$: PRINT
745 X$ = "000000.00": CALL 27904Z
     $: IF  VAL (Z$) < 1 THEN  PRINT
     A$(0)" ";: GOTO 790
750  FOR X = 0 TO 1:S = 0:Z = 0: FOR
     Y = 0 TO 2: IF Z > 9 THEN 780
755 Z =  VAL ( MID$ (X$,X * 3 + Y
      + 1,1)):S = S + Z: IF Z = 0
      THEN 780
760  IF Y <  > 1 THEN 775
765  IF Z <  > 1 THEN  PRINT B$(Z
     )" ";: GOTO 780
770 Z = 10 +  VAL ( MID$ (X$,X *
     3 + 3,1))
775  PRINT A$(Z)" "; IF Y = 0 THEN
      PRINT B$(0)" ";
780  NEXT : IF S > 0 AND X = 0 THEN
      PRINT B$(1)" ";
785  NEXT
790  PRINT "AND " RIGHT$ (X$,2)"/
     100"
795  PRINT : GOTO 700
1000  DATA ZERO,ONE,TWO,THREE,FOU
     R,FIVE,SIX,SEVEN,EIGHT,NINE,
     TEN,ELEVEN,TWELVE,THIRTEEN,F
     OURTEEN,FIFTEEN,SIXTEEN,SEVE
     NTEEN,EIGHTEEN,NINETEEN,HUND
     RED,THOUSAND,TWENTY,THIRTY,F
     OURTY,FIFTY,SIXTY,SEVENTY,EI
     GHTY,NINETY
```

Mr. Monclova is a systems programmer for a large, diversified computer user in Manhattan. You may contact Mr. Monclova at 229-03 129 Ave., Laurelton, NY 11413.

MICRO™

A Full Byte for Your APPLE Printer

by Mark J. Boyd

Many modern printers use an 8-bit code, but most Apple printer interface cards can send only 7-bit ASCII code. This article shows an easy way to add program control of the eighth bit to allow full use of your printer.

Full Byte
requires:
Apple II, a printer,
and a one-wire modification

This article is for the many Apple users who have a printer connected to their computer by an interface that sends only 7-bit ASCII codes. Specifically, I discuss the Epson MX-80 and the Epson Apple interface, but other interfaces can be modified in a similar manner, and other printers can make use of the full 8-bit code. For the MX-80, use of bit 8 means block graphics and full control of line spacing.

To gain control of bit 8, which is normally held low by the interface card, you must find a TTL output that can be controlled from your programs. The Apple II has four such outputs, called annunciator outputs, located at the game I/O connector. Two of these outputs, AN0 and AN1, are set low on power-up or reset. This is just what you need, since bit 8 should be low for normal use of the printer.

You must connect the annunciator output to bit 8 on the printer interface. On the Epson interface card this bit is normally held low by a jumper to ground. All you need to do is disconnect it from ground and connect it with a piece of wire to the annunicator output. The wire can be easily routed from the interface card to the game I/O socket without interferring with other peripheral cards.

The annunciator outputs are controlled by accessing certain locations in memory. There is no actual memory at those locations, but trying to read or write it sets what Apple calls "soft switches" (TTL flip-flops). For AN1, which I used, the locations are −16293 (to set it high) and −16294 (to set it low). When sending a control sequence to set the printer for a spacing of N, 1/72-inch increments per line, use the following code:

```
? CHR$ (27) "A";: POKE −16293,0:
? CHR$ (N);: POKE −16294,0:? CHR$
(27) "2"
```

The first POKE sets bit 8 high. This causes the interface to actually send CHR$ (N+128), as required by the printer. The second POKE sets bit 8 low again so the second ESC sequence is sent properly.

The best way to make the connection is by soldering a 16-pin DIP header to a 16-pin DIP socket. The wire to the interface then can be soldered to the AN1 connection (pin 14) between the socket and the header. Pin 14 is the third pin up on the left-hand side of the socket when viewed from the front of the Apple. This method leaves the game I/O connector free for other use. A simpler way, and the one I used, is to solder a small-diameter insulated solid wire to the interface connection (the non-grounded side of the jumper on the Epson; the left-hand side when viewed component side up). The other end of this wire can be plugged directly into the game I/O socket. You can avoid soldering by using a jumper wire with insulated miniature alligator clips. Clip one end to the non-grounded side of the jumper after you cut it, and clip the other end to a short piece of bare wire (a ¼ w resistor lead will work) inserted into the game I/O socket. Be very sure the alligator clips cannot come into contact with any other circuits. If you use either of these latter two methods, you will have to remove or bend out the corresponding pin on any game I/O plugs you use.

After deciding on the method to use and attaching the wire to the interface card, you should check this connection before making the connection to the game I/O socket. Replace the interface card in your Apple and connect the

***Listing 1:* Modifications to Taylor's Routine**

```
*942B:  20 A4 94                   ; JSR to new setup location
*955D:  20 9D 94                   ; JSR to offset routine
*958E;  20 9D 94                   ; JSR to offset routine
*949D:  18 69 20                   ;
*94A0:  20 02 C1 60 A2 C1 A9 09    ; New routine, adds 32 to
*94A8:  9D B8 06 A9 FF 9D B8 04    ; Graphic character value
*94B0:  9D 38 06 A9 00 9D B8 05    ; Turn on interface, etc.
*94B8:  20 EA 03 A9 0F 20 02 C1    ; (as in Taylor's routine)
*94C0:  8D 5B C0 60                ; INIT printer for compressed
                                     Print and set bit 8 high
```

printer. When you power up your system, leave the other end of the new wire unconnected. Turn on the printer, enable the interface, and try printing some text. It should print as graphics characters. Now ground the wire by holding the stripped end against the power supply. Text should print normally. If this connection is made properly, you cannot harm the Apple by plugging it into any of the game I/O sockets. If it is not connected properly (say you connected to the ground side instead of the bit-8 side of the cut jumper) you might damage the chip that drives the annunciator outputs.

Once you are convinced that everything is OK at the interface end, turn off the Apple and make the connection to AN1. Now turn on your system and print some text. Everything should work normally. If not, shut it down and check your connections before proceeding.

Next POKE – 16293,0 to switch bit 8 high. Print a line of text to verify that the printer is printing graphics characters. Finally, POKE – 16294,0 to set bit 8 low again. Everything should be back to normal and you now have a full 8-bit interface for your printer!

Now use the interface to print the HGR screen on the MX-80 (other systems with TRS-80-compatible graphics also will work, but probably you will have to change the offset value for graphics character set and the printer initialization). Charles Taylor's article, "Apple Graphics for Okidata Microline 80," (MICRO 48:48) inspired me to get control of bit 8 on my interface. His assembly-language program needs only minor modification for use with program control of bit 8 (he used a switch) and the MX-80.

Enter Mr. Taylor's assembly-language program in hexadecimal, using the Apple monitor memory-load function (page 44 in the *Apple II Reference Manual*.) You can check your entries most easily with the L command. Since the entries must be correct and entry is tedious, check things as you go along. It's terrible to finish entering a few hundred hexadecimal pairs of numbers only to discover you entered an extra byte early on.

After you have entered and carefully checked Mr. Taylor's program, make the entries shown in listing 1: relocate the start of the printer setup routine to make room for the offset routine; replace the setup routine with the offset routine, followed by the new setup routine; and redirect the graphics output characters through the offset routine. I couldn't find enough room to add a printer/computer reset routine without reassembling Mr. Taylor's program, so you'll have to reset both the computer and the printer after running it.

After checking your entries again (you can't be too thorough when checking machine-language entries), BSAVE the routine (A$ 9400, L$ 196). The program can print either screen (see Taylor's article), so get a screen or two into memory and check it out. Turn your printer off and set the paper to top of form before running. The program turns on the interface (in slot 1), so all you need to do is set the paper correctly, turn on the printer, and run it. A screen fills a normal-sized page and takes about five minutes to print.

Before you entered Taylor's program, you were told to set HIMEM to 37760. In the future, before you BLOAD it you must always set HIMEM. If you plan to call this routine from a BASIC program you should set HIMEM and BLOAD it first, then load the calling program. The calling program should reset the printer (turn off compressed print, reduce the line width back to normal) and set bit 8 low after calling the routine.

For stand-alone use, I am including a couple of BASIC programs that allow you to "EXEC SP" and follow the instructions given on the screen. The first program, SP CREATE, should be loaded and run to create an EXEC File, SP. SP sets HIMEM, BLOADs SPRINT (my name for the screen printing routine), and RUNs SCREEN PRINT. SCREEN PRINT takes care of screen selection, displays the chosen screen, allows you to print a title above the screen, and resets the printer and computer when it is finished.

Mark Boyd teaches at a small four-year liberal arts college. His interest in computers began when he was an undergraduate taking a FORTRAN programming course. You may contact Dr. Boyd at St. Mary of the Plains College, Dodge City, KS 67801.

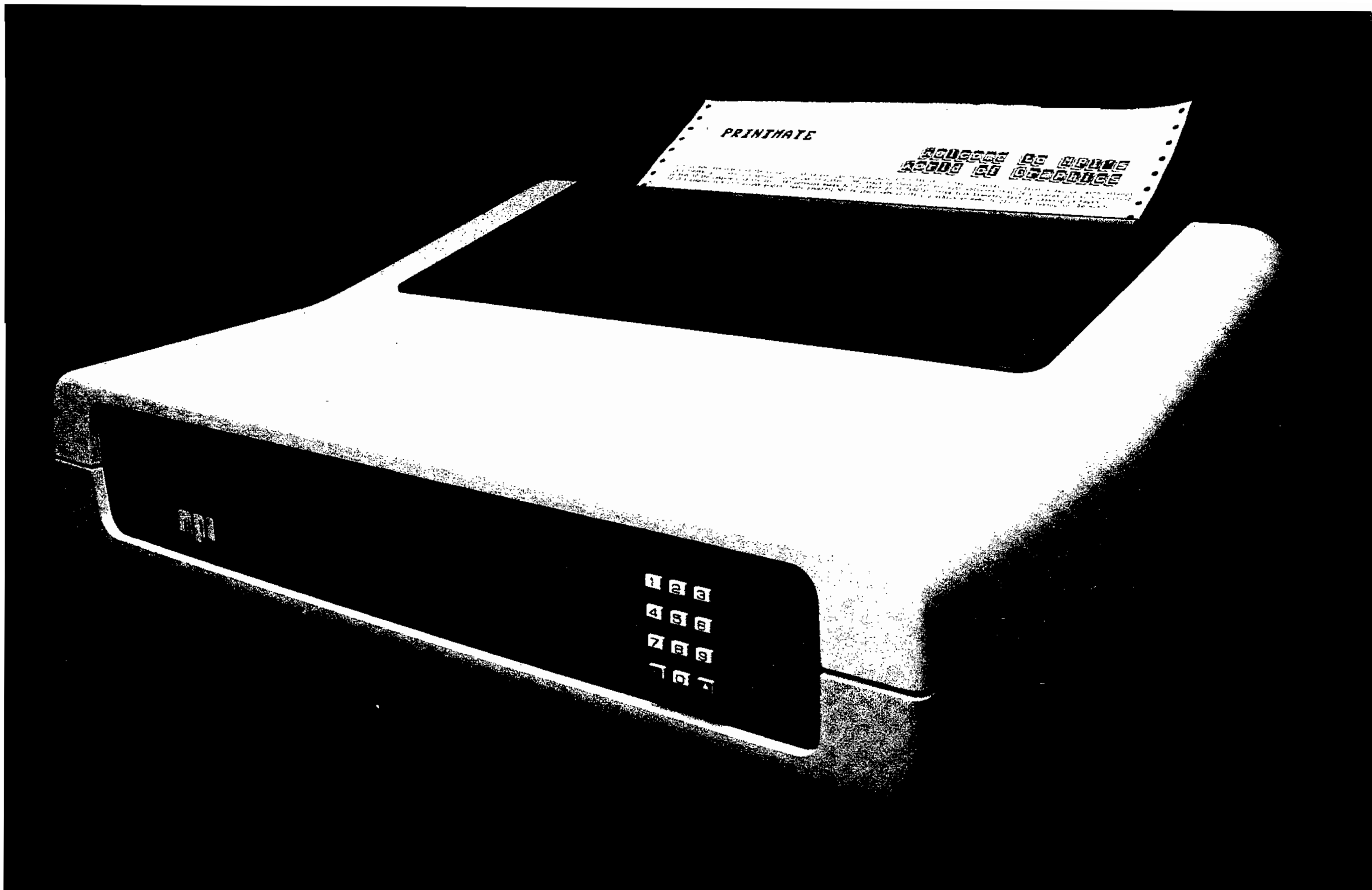
MICRO

***Listing 2:* EXEC File Maker**

```
10   D$ = CHR$ (4)
20   PRINT D$"OPEN SP"
30   PRINT D$"WRITE SP"
40   PRINT "HIMEM: 37760"
50   PRINT "BLOAD SPRINT"
60   PRINT "RUN SCREEN PRINT
70   PRINT D$"CLOSE SP"
80   END
```

***Listing 3:* Screen Print Listing**

```
4  REM  SETUP CTRL-D,CTRL-I, AND TURN OFF INTERFACE BOARDS
5 D$ =  CHR$ (4):P$ =  CHR$ (9): PRINT D$"PR#0"
10  TEXT
20  HOME : REM  CLEAR SCREEN
30  PRINT  TAB( 4);"SCREEN PRINT PROGRAM - PLEASE TURN"
31  PRINT  TAB( 12);"OFF THE PRINTER"
40  PRINT "LINE UP THE TOP OF THE FORM (PERFORATIONS) WITH THE PRINT HEAD.
     NOW TURN THE PRINTER ON AND MAKE SURE THE READY LIGHT IS ON.
43  FOR I = 1 TO 5000: NEXT : REM  WAIT FOR USER TO READ PROMPT
45  PRINT  PEEK ( - 16297); PEEK ( - 16302): REM  SETUP FOR HGR
49  HOME
50  INPUT "WHICH SCREEN DO YOU WANT? (1/2)";A$
52  IF A$ = "2" THEN 54
53  PRINT  PEEK ( - 16300): GOTO 55: REM   SELECT SCREEN 1
54  PRINT  PEEK ( - 16299): REM  SELECT SCREEN 2
55  PRINT  PEEK ( - 16304): FOR I = 1 TO 2000: NEXT : TEXT : REM  DISPLAY
    SCREEN
56  HOME : INPUT "IS THIS THE SCREEN YOU WANT? (Y/N)";E$
57  IF E$ <  > "Y" THEN 49
60  INPUT "DO YOU WANT A TITLE PRINTED? (Y/N)";B$
70  IF B$ <  > "Y" THEN 90
80  INPUT "PLEASE ENTER THE TITLE.";C$
90  HOME
100  IF B$ <  > "Y" THEN 130
110  PRINT D$"PR#1": PRINT P$"80N": REM  TURN ON THE PRINTER INTERFACE AND
      SET FOR 80 COL
120  PRINT  TAB( (80 -  LEN (C$)) / 2);C$: REM  PRINT CENTERED TITLE
130  IF A$ <  > "1" THEN 150
140  CALL 37888: REM  PRINT SCREEN 1
150  IF A$ <  > "2" THEN 170
160  CALL 37904: REM  PRINT SCREEN 2
170  POKE  - 16294,0: PRINT  CHR$ (127): PRINT P$"I": PRINT D$"PR#0": REM
       RESTORE NORMAL SETTINGS
180  END
```



PrintMate™ 150

THE MOST ADVANCED PRINTER IN ITS CLASS.

Good news for microsystem and personal computer users! MPI offers four wide carriage printers with excellence in price and performance. The two "A" versions of PrintMate™ 150 feature a factory installed "SoftSwitch™" front panel keypad, with a 4K buffer on PrintMate™ 150 model A1 and a 16K buffer on model A2. PrintMate™ 150 models B1 and B2 are factory equipped with a 2K and 16K buffer, respectively. PrintMate™ 150 models have an exceptional set of outstanding graphics and font capabilities, optional expansion, and other advanced features that differentiate the PrintMate™ 150 from its competitive rivals as the superior performer. A bold claim? The strong and widespread acceptance of the excellent PrintMate™ 150 is based on outstanding user features:

HIGH SYSTEM THRUPUT—150 characters per second advanced logic seeking impact printing with an accelerated print head slew rate and turnaround makes PrintMate™ 150 a high speed performer.

WIDE CARRIAGE VERSATILITY—The PrintMate™ 150's wide carriage can accommodate print lines from 136 to 231 characters in length and can easily handle forms from 3 to 15 inches wide and as long as 31 inches.

LARGE SELECTION OF PRINT CAPABILITIES—The 7x9 dot matrix allows user selection of 10, 12, 15 or 17 characters per inch or the 11x9 serif font provides document quality printing at 10 characters per inch.

"SoftSwitch™" FRONT PANEL CONTROL—The PrintMate™ 150 A models have SoftSwitch™" front panel keypads for externally changing forms length, print density, horizontal and vertical tabs, baud rate and character set. A simple "SoftSwitch™" entry will display the operating mode you have selected and PrintMate™ 150 responds to every entry with a pleasant tone of confirmation. With the "SoftSwitch™", you can turn off the printer—even unplug it—and PrintMate™ 150 will retain every detail in its non-volatile memory. The "SoftSwitch™" may be added to the PrintMate™ B models.

EXPANDABLE PRINT BUFFER—PrintMate™ 150 models A2 and B2 have a factory installed 16K buffer. Both the 4K buffer model A1 and the standard 2K buffer model B1 are optionally expanded in increments to 16K. The PrintMate™ 150's expanded buffer allows application extensions for high speed interleaved printing and spooling, greatly improving the host computer's performance in applications that are print bound.

DOWNLINE LOADABLE FONTS—The powerful microprocessor based command set of the PrintMate™ 150 allows a custom character set to be developed in the host computer and downloaded to any PrintMate™ 150 model with a 4K or larger buffer.

GRAPHICS—The standard graphics capabilities of all PrintMate™ 150 models allow printing of up to 6,120 individually addressable dots per square inch giving exceptional resolution for graphics and special characters.

PrintMate™ APPLICATIONS PACKAGES—Turn-key graphics and display fonts can be implemented with an extensive line of MPI supported and maintained AP-PAK™ applications packages providing specialized fonts, custom graphs, tables, and picture graphics. Specialized characters such as logos may be easily defined and edited for printing directly from your computer.

CONSIDER THE FEATURES—Only PrintMate™ 150 offers so many ways to get your message across: graphics; display fonts; downline loadable character sets; high print speed; advanced logic seeking; 15 inch wide carriage; a variety of forms and paper capabilities and; friendly "SoftSwitch™" interaction. The PrintMate™ 150 is the responsive performer that perfectly mates with your microsystem or personal computer. With prices beginning at $995, it is evident that the PrintMate™ 150 is the superior performe in function and price.

The Printer People

mpi™

Micro Peripherals, Inc.
4426 South Century Drive
Salt Lake City, UT 84107
Phone 1-800-821-8848

Circle No. 31

The MICRO™

Atari 800 | Texas Instrument

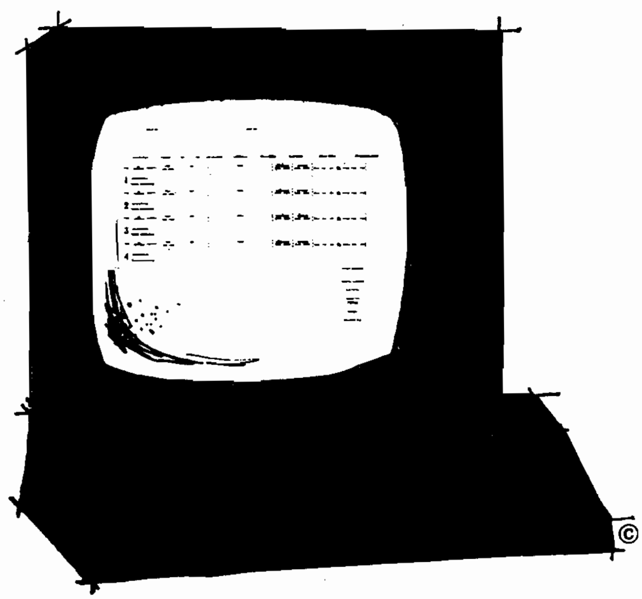
LEARNING CENTER

Features:

- **BANNER: A Display Program for the Color Computer**
  by Bryan Christiansen
  This "ticket-tape" program, written by a 15-year-old high school freshman, displays any message of your choice on a scrolling screen.

- **DIGI-DRAFT**
  by Tim Kilby
  "DIGI-DRAFT" is a graphics program that allows images to be drawn on the screen with convenient commands

- **The Computer Revolution Reaches Out to the Community**
  by Emmalyn H. Bentley
  A look at the new Microcomputing Learning Center in Nashua, N.H. — who is enrolling, what courses are offered, and how the Center is contributing to the community as a whole.



- **MICRO Calc for the VIC-20**
  by Loren Wright
  A computer worksheet program that allows you to define a series of calculations and preform them at the touch of a key.

PUT THE FULL POWER OF YOUR VIC-20 AT YOUR COMMAND!

Order your copy of MICRO's newest book...

MASTERING YOUR VIC-20

with eight BASIC projects

You'll Receive:

- **MICRO Calc...** a miniature spreadsheet program that makes complex, repetitive calculations a breeze.
- **MASTER...** a guessing game that teaches programming with random numbers and flags.
- **VIC Clock...** to teach you ON..GOSUB function and character graphics.
- **BREAK-UP...** a popular game that also teaches how animation is achieved with PEEKS and POKES to screen memory.
- ***Plus...*** music programming, string manipulation, sorting demonstrations, and more.

Use this coupon or the postage paid card in this issue to order.

MICRO Books P.O. Box 6502 Chelmsford [illegible]

[illegible] YES, please rush [illegible] copies of MASTERING YOUR VIC-20 [illegible]
at only $[illegible] per copy plus [illegible]

My payment of [illegible]

NAME ____________

ADDRESS ____________

CITY ____________ [illegible]

CREDIT CARD # ____________ [illegible] DATE ____________

Each Program Worth the Price of the Book!

Order your copy of MASTERING YOUR VIC-20 Today!

MICRO Calc for Commodore and APPLE

by Loren Wright

Minor changes provided for Commodore-64 and all PET models. Equivalent program for Apple computers is presented on page 53.

What is MICRO Calc?

The "electronic spread-sheet" is one of the most popular types of business programs available. The first program of this kind, "VisiCalc" (now sold by VisiCorp), has been credited with Apple's big impact on the business market. A spread-sheet program lets you perform mathematical computations on the video display. You enter formulas and data directly into the display *via* the keyboard. Commercial spread-sheet packages offer a large array of cells into which you can program labels or values. The values can be defined in very complex ways, including values calculated in other cells. The applications range from accounting to inventory to printing bar graphs. Once your sheet is defined for a particular application, you can save it and use it again and again for that application.

MICRO Calc is a miniaturized version of one of these spread-sheet programs. Instead of cells, you have ten lines on the screen with which to work. MICRO Calc is a short program that allows you to perform even very elaborate calculations at the touch of a key. In addition, MICRO Calc can be used to learn how BASIC functions work. The format is similar to how BASIC programs themselves are written.

As presented, MICRO Calc runs in an unexpanded VIC. The minor changes described allow the program to take advantage of more memory in the VIC, to use a disk drive, or to run on the PET or Commodore 64. For the Apple, use the equivalent program by Phil Daley on page

Entering the Program

Type the program in exactly as it is shown in listing 1. Do not include any spaces unless they appear within quotes. If your machine is not a VIC, be sure to make the changes described for your machine at the end of this article. When you have finished typing, check carefully against the listing, correct any errors, and SAVE the program to a cassette (if you don't have a cassette you'll have to retype this program for every session you want to use it!). Rewind the cassette, VERIFY it, make sure ST = 0, and you are ready to try it out.

Using MICRO Calc

If you don't already have MICRO Calc in memory, LOAD it from the tape. Type RUN and press RETURN. After a brief delay, the screen will fill with ten yellow lines and a yellow diamond will appear in the upper left-hand corner. The diamond, called the "cursor," indicates where characters you type will appear on the screen.

A Simple Example

For this first example we want to add two numbers and print the result. Our two numbers are labeled A and B and the sum defined as C. Type the left-hand portion of each of the following lines, pressing RETURN after each. The rest of the line is printed here only to help you understand what is going on.

```
A=5         ;assign 5 to A
B=7         ;assign 7 to B
C=A+B       ;assign sum to C
C?          ;request value of C
```

Now press '@'. The cursor disappears for a second, the answer '12' (the sum of 5 + 7) is printed after the 'C?' and the cursor reappears after the '5' on the first line. To change the value of A to equal 2, press the INST/DEL key, type '2', and then '@'. The new answer (9) appears after the 'C?'. The formula may be changed. For example, to make it 'C = A – B', use the CRSR down key (unshifted) to move the cursor to the end of the third line. Remove the ' + B' by pressing INST/DEL twice, then type the new portion of the formula ' – B'. When you press '@', the new answer, ' – 5', appears on the fourth line.

Now let's try a more practical example — calculating bowling averages. Use shifted CLR/HOME to clear the screen, and type the following, pressing RETURN after each line.

```
A=165       ;first score in A
B=148       ;second score in B
C=173       ;third score in C
S=A+B+C     ;sum of three scores
S?          ;display sum
V=S / 3     ;calculate average score
V?          ;display average score
```

Press '@' and, after a brief delay, the sum of the three scores, '486', will appear after the S? on the fifth line and the average of the scores, '162', after the V? on the seventh. Of course this procedure is not restricted to bowling scores. Any three numbers you want to average could be used.

To show how similar "programming" MICRO Calc is to writing BASIC, here is a BASIC program that does the same thing as the averages sheet above.

```
10 INPUT"FIRST SCORE";A
20 INPUT"SECOND SCORE";B
30 INPUT"THIRD SCORE";C
40 S=A+B+C
50 PRINT S
60 V=S / 3
70 PRINT V
80 GOTO 10
```

The Rules

MICRO Calc, as you can see, is easy to use. However, it has some rules that must be followed to avoid disastrous results.

1. Each line must begin with a letter, followed either by a '?' or a '='.
2. Nothing should be typed after a '?'. That is where the program fills in its result.
3. After a '=', you must type either a number or a BASIC expression that evaluates to a number. Any BASIC function that yields a numerical result, such as TAN, SQR, RND, LOG, or ABS, may be used. BASIC string functions, such as LEFT$, MID$, ASC, or VAL, and integer variables may not be used.

Consult your "VIC User's Guide" to learn about BASIC's built-in functions. Then use MICRO Calc to help you understand how they work.

MICRO Calc allows you to use 26 storage locations called "variables." Each variable is uniquely identified by a letter of the alphabet. In our first example, when we typed 'A=5', we assigned the value of 5 to the variable A, we typed 'B=6' to assign the value 6 to B, and typed 'C=A+B' to assign the sum of A and B to C. Until it is changed, 5 is substituted for A whenever it is used in an expression. Likewise, B and C retain their values until they are changed.

As in a BASIC program, MICRO Calc looks at the lines on the screen from the top down. Therefore, a 'C?' on a line before C is defined will give the wrong answer. Also, if you assign a variable a new value, the new value will be used in all subsequent calculations.

Editing

MICRO Calc has limited editing capabilities. You can move the cursor to the end of the line below by pressing either RETURN or CRSR down. From the last line, the cursor will move to the top line. Similarly, CRSR up will move the cursor to the end of the line above, and from the top line to the bottom line. The only way to change an existing line is to delete from the right end and retype.

MICRO Calc Summary

Command Characters

@	Calculate
INST/DEL	Delete previous character
CRSR dn	Move to end of next line
CRSR up	Move to end of last line
RETURN	Move to end of next line
←	LOAD or SAVE screen
CLR/HOME	Clear screen lines

Characters Allowed

A..Z 0..9

/ * + − = . ( ) >< ↑ ?

Optional Characters (program lines must be added)

! for remark

: to separate multiple statements on a line

space

Avoiding Errors

To alleviate the chance for problems, MICRO Calc disallows characters that could cause trouble. However, if you are not careful it is easy to make a fatal error. The program will stop and an error message, such as ?SYNTAX ERROR or ?DIVISION BY ZERO ERROR, will be printed. Should one of these occur, clear the screen, type RUN, and press RETURN. You will have lost everything that was on the screen, but you shouldn't have to reload the program from cassette.

If you use parentheses to assign a variable, for example 'C=(A+B)/(A−B), be sure there are as many right parens as left parens. Also, remember that, unlike in algebra, multiplication here must be explicitly indicated with an asterisk. Use '10*B' and '5*(A+B)', not '10B' and '5(A+B). If you use a variable that has not been assigned in a previous line, its value is assumed to be zero. Therefore, division by an unassigned variable will result in a ?DIVISION BY ZERO ERROR.

Saving the Screen

To save a format for re-use, type in just the assignments (A=, B=, ...) and the formulas (C=A+B), and then save to tape without running the program. For example, clear the MICRO Calc screen using shift CLR/HOME, and type in the second example (bowling average) without any values for A, B, and C. Don't press '@'! The program would crash, but this is the most convenient form for saving the screen. When you are satisfied you have typed the screen correctly, save the screen to tape. On the VIC-20, press the ' ' key and then press 'S' for SAVE. Type in 'AVERAGE', position the tape, and press RETURN. Your screen is now stored on tape under the name you entered.

Loading the Screen

Clear the MICRO Calc screen using shift CLR/HOME, and press the ' ← ' key. This time press 'L' for LOAD, position the tape, enter 'AVERAGE', and press RETURN. If all goes well, in a few seconds the screen will return with the lines you saved before.

More Examples

According to the Pythagorean Theorem, "The hypoteneuse of a right triangle is equal to the square root of the sum of the squares of the other two sides." You can use MICRO Calc to solve right triangles quickly. Enter the following lines, pressing RETURN after each line.

```
A=3                 ; Assign one side equal to 3
B=4                 ; Assign another side equal
                      to 4
H=SQR(A↑2+B↑2)      ; The equation
H?                  ; The answer
```

You can calculate monthly payments of installment loans using the following formula:

$$D = \frac{1 - \frac{1}{(1+i)^m}}{i}$$

i is the interest rate per month and *m* is the number of months. The principal (the amount you are borrowing) is divided by *D* to get the monthly payment. Below is the screen you enter to perform these calculations with MICRO Calc.

```
A=                  Principal
M=                  Number of Months of the Loan
I=                  Interest Rate (Annual
                    Percentage)
I=I/1200            Monthly Interest Rate (decimal)
D=(1-(1+I)↑-M)/I    Calculate Divisor
P=A/D               Calculate Monthly Rate
P=INT(P*100+.5)/100 Round to Nearest Cent
P?                  Display Payment
```

Before you try it out, SAVE the screen as described above. When it is SAVEd the program will return with the cursor at the end of what you typed on the first line. Enter values for A, M, and I, press '@' and the result will appear on the eighth line. Now you can make a change, as you did in the example above; hit '@' and instantly see the change in the monthly payment.

Once you have the hang of MICRO Calc, you can use it to perform a wide variety of repetitive calculations. Be sure to save the more elaborate screens to tape, and you will develop a library of useful screens.

Improving MICRO Calc

MICRO Calc was intentionally kept simple so that it would be easy to type in and easy to use. The next section describes how you can make changes in a few lines to improve the power of MICRO Calc.

Flashing cursor:

With the MICRO Calc program in memory type the following program lines:

```
4500 POKE 204,0
4510 GETT$:IFT$="  "THEN 4510
4520 POKE 204,1: RETURN
```

For the PET use 167 instead of 204 in lines 4500 and 4520. Normally, when the GET function is used, the cursor is turned off. POKEing a 0 into this location turns the flashing cursor on, and POKEing a 1 turns it off. Incidentally, you can change the cursor character from the diamond to anything you want. Just change the diamond near the end of line 20 to the desired character. If you want to save this, or any other enhanced version, be sure you don't record over the original version of the program.

Cursor Control Characters in Listings

Commodore computers allow cursor controls to be programmed as characters within a string. However, in listings these appear as reverse field characters (for instance, 'clear screen' is represented by a reversed heart character). To make these listings easier to type in and easier to understand, we have substituted bracketed abbreviations of the functions of these characters. When more than one appears in a row, the number of repeats precedes the abbreviation (i.e., [2 CU] means hold shift and press CRSR up/dn twice).

Expression	Function	Key(s)
[CLR]	Clear screen	shift CLR/HOME
[HOME]	Home cursor	CLR/HOME
[CD]	Cursor down	CRSR up/dn
[CU]	Cursor up	shift CRSR up/dn
[CR]	Cursor right	CRSR left/right
[CL]	Cursor left	shift CRSR left/right
[RVS]	Reverse field	OFF/RVS
[OFF]	Reverse off	shift OFF/RVS
[SPC]	Space	Space bar
[YEL]	Yellow	CTRL 8

Twenty working lines:

If you have particularly lengthy screens, you may need more than the ten lines provided. Upgrading to twenty lines is easily accomplished by using the lines in between the original ten. Several program lines must be changed to accomplish this:

```
30 NL = 20:DIMS$(NL),S(NL)
230 PRINTRB$CR$;:GOTO110
270 S$(LL) = S$:LL = LL – 1:IFLL = 0THENLL = NL:
    PRINTRB$"[CH][20 CD]";:GOTO110
280 PRINTRB$CR$"[2 CU]";:GOTO110
7020 PRINTCR$S$X$:NEXT:RETURN
8520 PRINT"[YEL]"S$"[RVS]"LEFT$(BL$,20 – LEN
    (S$)):NEXT:PRINT"[CH][CD]";:RETURN
```

Note that the most significant change is accomplished by merely replacing the value of NL in line 30. NL is used in lines 220, 270, 3000, 5090, 5110, 7000, and 8500. With twenty lines, the calculations will take longer.

Comments and Multiple Statements:

For some applications, you may want to type several assignments on one line or include a comment for documentation. Change the following lines to add these features:

```
2020 IFT$ >",''ANDT$ < ";"THEN2070
2050 IFT$ = "[SPC]"ORT$ = "!"ORT$ = " ↑ "
    THEN2070
3015 IFLEFT$(S$(JJ),1) = "!"THEN3090
4000 II = 0:KK = II
4010 II = II + 1:KK = KK + 1:IFII > LEN(A$)
    THENGOSUB4100:RETURN
4020 XX = ASC(MID$(A$,II,1)):IFXX = 58
    THENGOSUB4100:GOTO4010
4030 IFXX = 33THENGOSUB4100:RETURN
4040 POKE511 + KK,XX:GOTO4010
4100 POKE511 + KK,0:KK = 0:SYS828:RETURN
```

The program will now allow you to type colons, exclamation points, and spaces. Everything appearing after a colon on a line will be ignored in evaluations. It is even possible to have a whole line as a comment. Also, assignment statements (but not value requests) may be put together on one line, separated by colons. While these features make MICRO Calc more powerful, they also make editing an existing screen more difficult. To make changes, you must still delete all characters back to the change. This means that comments must be retyped each time a change is made earlier in the line. See the section marked "Suggestions for other changes:" for some tips on improving MICRO Calc's editing.

Saving files on disk:

Enter the following lines to substitute disk storage for cassette storage:

```
5020 IFT$ = "L"THENSA = 9:F$ = ",S,R":
    GOTO5045
5030 IFT$ = "S"THENSA = 10:F$ = ",S,W":
    GOTO5045
5045 PRINT"DRIVE NUMBER ([RVS]0[OFF] OR
    [RVS]1[OFF])"
5046 GOSUB4500:IFT$ < "0"ORT$ > "1"THEN5046
5050 INPUT"[2 CD]NAME";NA$:NA$ = T$ + ":"
    + NA$ + F$
5060 OPEN1,8,SA,NA$:IFSA = 10THENGOSUB5090:
    GOTO5080
```

The major changes are selecting the drive number, changing the primary and secondary addresses, and adding the prefix and suffix to the file name. The actual reading and writing process is exactly the same as for cassette.

Suggestions for other changes:

With just a little work, MICRO Calc can be made to handle strings as well as numbers. First, all string names within the program must be changed to two characters. Second, a separate series of subroutines with an appropriate calling routine, similar to 6500-6810, must be added to extract the proper values for printing. Third, the quote and dollar characters must be accepted by the editor. Finally, the print routine at 7000 must be made to recognize and print strings. Probably other minor problems would have to be solved as well. The editor can be improved considerably by allowing insert, delete, and cursor movements within a line. With larger screens, it may be desirable to divide some of the lines into smaller cells.

How the Program Works

If you examine listing 1, you will notice that all floating point variable names have two characters. That is because the single-character names A-Z are reserved for the program user. The key to operation of the program is the machine-language program contained in the DATA statements 9828-9868. Listing 2 is an assembly listing (for the VIC) of this routine. The heart of the routine is two JSRs to the BASIC ROM routines LET and TKENIZE. The rest is involved with manipulating the character pointer. Using this little bit of machine language saves countless lines of BASIC programming.

The routine is POKEd into memory as soon as the screen is cleared and the screen color set to black (line 10). The next two lines define constants and dimension the two arrays used to hold the contents and values for each line of the MICRO Calc screen. Using constants instead of literal values does more than save space in the program. It also speeds execution (BASIC doesn't have to figure out what 10 means every time, when it can look it up under NL) and makes changes easy. Line 100 starts the mainline of the program. The line pointer is set to the first line and the screen is printed using subroutine 8500. Line 110 is where the loop returns to whenever the cursor is moved to a new line. The string array element containing the contents of the current line is moved into a temporary location. Then the line is printed, followed by the cursor (DI$). Subroutine 2000 GETs characters from the keyboard, accepts or rejects them, and returns on an acceptable character. Some characters, checked in 2010, are control or editing characters. These are dealt with in line 130-180 in the main program. Other acceptable characters are printed, with a return to the main program where they are added to the temporary line string in 190. In line 200, the length of the string is checked. If the last character fills the line, the temporary string is stored in that line's array element, and the cursor automatically advances to the next line.

Lines 210-230 handle advancing to the next line, lines 240-260 handle a delete, and lines 270-280 handle moving up one line. Line 130 handles the '@' or 'calculate' directive. First the cursor is removed by printing RB$. Then the line contents are stored in the proper array element. Next the evaluation (3000) and printing (7000) routines are called. Finally the cursor is returned to the end of the first screen line.

The evaluation subroutine (3000-3030) sets all the single-letter variables equal to zero (subroutine 9000). Again, using A instead of zero saves a little execution time. If the last character is a '?', then subroutine 6500 is used to assign the proper variable value to the value array element for that screen line. Ideally, the ON...GOSUB structure should have all 26 routines listed on one line. However, because the VIC allows only 88 characters on a program line, we have to split it up. This is accomplished by converting the codes for the letters A-Z to numbers 1 to 26. If the resulting number is 13 or less, then line 6510 is used. Numbers 14 to 26 are converted to 1 to 13, and line 6530 is used. If the last character is not a '?', then the statement is assumed to be an assignment (A=, Z=, etc.). Subroutine 4000 POKEs the characters of the line one-by-one into the BASIC input buffer and sticks a zero at the end. The machine-language routine described above is called with SYS828, and a return is made to line 3030.

Subroutine 3000 continues through the array of screen lines, handling value requests (A?) one way and handling assignments (A=) another. Only subroutine 5000 is left. The LOAD/SAVE routine is invoked whenever the '   ' key is pressed. First a choice is offered between LOAD and SAVE. Then the file name is entered. Subroutine 5090 is used for SAVEing and subroutine 5110 for LOADing. When a screen is saved, each element in the line array S$( ) is added to the string A$, along with a carriage return. If the array element is empty it is filled with a dummy character. The carriage returns are used to separate the items, so they can be read back with the INPUT# statement in line 5110. The dummy characters are discarded and replaced with empty or "null" strings.

Before returning to the main program, the machine-language routine must be POKEd back into the cassette buffer (8000). It is destroyed on any LOAD or SAVE operation. Finally, subroutine 8510 is called to reprint the screen. Note that a call to 8500, as in line 100, prints an empty screen, while starting at 8510 does not clear the arrays.

MICRO Calc Variable Usage

Constants

NL	number of screen lines
CR$	carriage return
DL$	delete character
RB$	reverse blank
BL$	twenty spaces
DI$	diamond cursor

Variables

A...Z	reserved for program user
II	loop counter
JJ	loop counter
LL	screen line pointer
XX	temporary, used in value passing
BB	letter of variable converted to index 1...26
SS	current element of value array
S$	working copy of screen line
T$	last character from GET (4500)
A$	working string for tape and disk
X$	used in formatting screen lines
S$( )	array of screen line contents
S( )	array of screen line values

Running MICRO Calc on Other Commodore Computers

MICRO Calc will run with relatively few changes on any Commodore machine. Provided here are substitute program lines for the Commodore 64 (listing 3), 4.0 PET (listing 4), 2.0 PET (listing 5), and 1.0 PET (listing 6). The only differences are in line 10 and lines 9828-9868. Type the lines appropriate to your machine, followed by the rest of listing 1. The program will now run as it is. However, you may want to take better advantage of your 40-column screen by changing the lines in listing 7.

Listing 1: MICRO Calc BASIC Listing (for VIC)

```
1Ø PRINT"[CLR]":POKE36879,8:GOSUB8ØØØ
2Ø CR$=CHR$(13):DL$=CHR$(2Ø):RB$="[RVS]
  [CL]":BL$="[2Ø SPACES]":DI$="♦[OFF][CL]"
3Ø NL=1Ø:DIMS$(NL),S(NL)
1ØØ LL=1:GOSUB85ØØ
11Ø S$=S$(LL):IFRIGHT$(S$,1)="?"THEN
   PRINT"[RVS]"BL$CR$"[CD]";
12Ø GOSUB2ØØØ
13Ø IFT$="@"THENPRINTRB$:S$(LL)=S$:
   GOSUB3ØØØ:GOSUB7ØØØ:PRINT"[CH][CD]";:
   LL=1:GOTO 11Ø
14Ø IFT$="[CLR]"THEN1ØØ
15Ø IFT$=CR$ORT$="[CD]"THEN21Ø
16Ø IFT$="[CU]"THEN27Ø
17Ø IFT$=DL$THEN24Ø
18Ø IFT$=""THENS$(LL)=S$:
   GOSUB5ØØØ:LL=1:GOTO11Ø
19Ø S$=S$+T$
2ØØ IFLEN(S$)<19THEN12Ø
21Ø S$(LL)=S$
22Ø LL=LL+1:IFLL=NL+1THENLL=1:
   PRINTRB$"[CH][CD]";:GOTO11Ø
23Ø PRINTRB$CR$CR$;:GOTO11Ø
24Ø IFS$=""THEN12Ø
25Ø PRINTRB$"[2 CL]"DI$;
26Ø S$=LEFT$(S$,LEN(S$)-1):GOTO12Ø
27Ø S$(LL)=S$:LL=LL-1:IFLL=ØTHENLL=NL:
   PRINTRB$"[CH][19 CD]";:GOTO11Ø
28Ø PRINTRB$CR$"[3 CU]";:GOTO11Ø
2ØØØ GOSUB45ØØ
2Ø1Ø IFT$="@"ORT$=CR$ORT$="[CD]"OR
   T$="[CU]"ORT$=""ORT$=DL$ORT$=
   "[CLR]"THENRETURN
2Ø2Ø IFT$>","ANDT$<";"THEN2Ø7Ø
2Ø3Ø IFT$>";"ANDT$<"["THEN2Ø7Ø
2Ø4Ø IFT$>"'"ANDT$<","THEN2Ø7Ø
2Ø5Ø IFT$="↑"THEN2Ø7Ø
2Ø6Ø GOTO2ØØØ
2Ø7Ø PRINTT$DI$;:RETURN
3ØØØ GOSUB9ØØØ:FORJJ=1TONL:IFRIGHT$(S$(JJ),1)
   ="?"THENGOSUB65ØØ:GOTO3Ø3Ø
3Ø1Ø IFLEN(S$(JJ))<3THEN3Ø3Ø
3Ø2Ø A$=S$(JJ):GOSUB4ØØØ
3Ø3Ø NEXT:RETURN
4ØØØ FORII=1TOLEN(A$):XX=ASC(MID$(A$,II,1)):
   POKE511+II,XX:NEXT
4Ø1Ø POKE511+II,Ø:SYS828:RETURN
45ØØ GETT$:IFT$=""THEN45ØØ
451Ø RETURN
5ØØØ PRINT"[CLR][RVS]L[OFF]OAD OR
   [RVS]S[OFF]AVE"
5Ø1Ø GOSUB45ØØ
5Ø2Ø IFT$="L"THENSA=Ø:GOTO5Ø5Ø
5Ø3Ø IFT$="S"THENSA=1:GOTO5Ø5Ø
5Ø4Ø GOTO5Ø1Ø
5Ø5Ø INPUT"[2 CD]NAME";NA$
5Ø6Ø OPEN1,1,SA,NA$:IFSATHEN
   GOSUB5Ø9Ø:GOTO5Ø8Ø
5Ø7Ø GOSUB511Ø
5Ø8Ø CLOSE1:GOSUB8ØØØ:GOSUB851Ø:RETURN
5Ø9Ø A$="":FORII=1TONL:S$=S$(II):IFS$=""THENS$="?"
51ØØ A$=A$+S$+CR$:NEXT:PRINT#1,A$:RETURN
511Ø FORII=1TONL:INPUT#1,A$:IFA$="?"THENA$=""
512Ø S$(II)=A$:NEXT:RETURN
65ØØ BB=ASC(LEFT$(S$(JJ),2))-64:
   IFBB>13THENBB=BB-13:GOTO653Ø
651Ø ONBBGOSUB656Ø,657Ø,658Ø,659Ø,66ØØ,661Ø,
   662Ø,663Ø,664Ø,665Ø,666Ø,667Ø,668Ø
652Ø GOTO654Ø
653Ø ONBBGOSUB669Ø,67ØØ,671Ø,672Ø,673Ø,674Ø,
   675Ø,676Ø,677Ø,678Ø,679Ø,68ØØ,681Ø
654Ø S(JJ)=XX
655Ø RETURN
656Ø XX=A:RETURN
657Ø XX=B:RETURN
658Ø XX=C:RETURN
659Ø XX=D:RETURN
66ØØ XX=E:RETURN
661Ø XX=F:RETURN
662Ø XX=G:RETURN
663Ø XX=H:RETURN
664Ø XX=I:RETURN
665Ø XX=J:RETURN
666Ø XX=K:RETURN
667Ø XX=L:RETURN
668Ø XX=M:RETURN
669Ø XX=N:RETURN
67ØØ XX=O:RETURN
671Ø XX=P:RETURN
672Ø XX=Q:RETURN
673Ø XX=R:RETURN
674Ø XX=S:RETURN
675Ø XX=T:RETURN
676Ø XX=U:RETURN
677Ø XX=V:RETURN
678Ø XX=W:RETURN
679Ø XX=X:RETURN
68ØØ XX=Y:RETURN
681Ø XX=Z:RETURN
7ØØØ PRINT"[CH]";:FORII=1TONL:
   S$=S$(II):SS=S(II)
7Ø1Ø X$="":IFRIGHT$(S$,1)="?"THENX$=STR$(SS)
   +"[RVS]"+LEFT$(BL$,16-LEN(STR$(SS)))
7Ø2Ø PRINTCR$S$X$:NEXT:RETURN
8ØØØ RESTORE:FORII=ØTO42:READAA:
   POKE828+II,AA:NEXT:RETURN
85ØØ FORII=1TONL:S$(II)="":S(II)=Ø:NEXT
851Ø PRINT"[CLR][CD]";:FORII=1TONL:S$=S$(II)
852Ø PRINT"[YEL]"S$"[RVS]"LEFT$(BL$,2Ø-LEN(S$))
   "[CD]":NEXT:PRINT"[CH][CD]";:RETURN
9ØØØ A=Ø:B=A:C=A:D=A:E=A:F=A:
   G=A:H=A:I=A:J=A:K=A:L=A:M=A
9Ø1Ø N=A:O=A:P=A:Q=A:R=A:S=A:T=A:
   U=A:V=A:W=A:X=A:Y=A:Z=A:RETURN
9828 DATA165,122,141,112,3,165,123,141,
   113,3,169,Ø,133,122,169,2,133,123,32,121
9848 DATA197,169,Ø,133,122,169,2,133,123,
   32,165,2Ø1,173,112,3,133,122,173,113,3
9868 DATA133,123,96
```

Listing 2: MICRO Calc Assembly Listing (for VIC). For Information only. Don't try to type this in.

```
0010 LET        .DE $C9A5            ;C64-$A9A5, 4.0-$B930
                0011 ;                          2.0-$C8AD
                0015 TKNIZE   .DE $C579         ;C64-$A579-  , 4.0-$B4FB
                0016 ;                          2.0-$C495
                0020 INBUFF   .DE $200
                0030 CHRPTR   .DE $7A           ;C64-$7A, -  2 & 4-$77
                0035 TEMP     .DE $0370
                0040 ;
                0050          .BA $33C
                0060          .OS
                0070 ;
033C- A5 7A     0080          LDA *CHRPTR       ;SAVE POINTER
033E- 8D 70 03  0090          STA TEMP
0341- A5 7B     0100          LDA *CHRPTR1
0343- 8D 71 03  0110          STA TEMP1
0346- A9 00     0120          LDA #L,INBUFF     ;POINT TO INPUT BUFFER
0348- 85 7A     0130          STA *CHRPTR
034A- A9 02     0140          LDA #H,INBUFF
034C- 85 7B     0150          STA *CHRPTR1
034E- 20 79 C5  0160          JSR TKNIZE        ;TOKENIZE -  STATEMENT
0351- A9 00     0170          LDA #L,INBUFF     ;POINT TO INPUT BUFFER
0353- 85 7A     0180          STA *CHRPTR
0355- A9 02     0190          LDA #H,INBUFF
0357- 85 7B     0200          STA *CHRPTR1
0359- 20 A5 C9  0210          JSR LET           ;VARIABLE -  ASSIGNMENT
035C- AD 70 03  0220          LDA TEMP          ;RESTORE POINTER
035F- 85 7A     0230          STA *CHRPTR
0361- AD 71 03  0240          LDA TEMP1
0364- 85 7B     0250          STA *CHRPTR1
0366- 60        0260          RTS
                0270          .EN
```

Listing 3: Revisions for Commodore 64

```
10 PRINT"[CLR]":POKE53281,0:GOSUB8000

9828 DATA165,122,141,112,3,165,123,141,113,3,
     169,0,133,122,169,2,133,123,32,121
9848 DATA165,169,0,133,122,169,2,133,123,32,165,
     169,173,112,3,133,122,173,113,3
9868 DATA133,123,96
```

Listing 4: Revisions for PET BASIC 4.0

```
10 PRINT"[CLR]":GOSUB8000

9828 DATA165,119,141,112,3,165,120,141,113,3
     169,0,133,119,169,2,133,120,32,251
9848 DATA180,169,0,133,119,169,2,133,120,32,48
     185,173,112,3,133,119,173,113,3
9868 DATA133,120,96
```

Listing 5: Revisions for PET BASIC 2.0

```
10 PRINT"[CLR]":GOSUB8000

9828  DATA165,119,141,112,3,165,120,141,113,3
      169,0,133,119,169,2,133,120,32,149
9848  DATA196,169,0,133,119,169,2,133,120,32
      173,200,173,112,3,133,119,173,113,3
9868  DATA133,120,96
```

Listing 6: Revisions for PET BASIC 1.0

```
10 PRINT"[CLR]":GOSUB8000

9828 DATA165,201,141,112,3,165,202,141,113,3
     169,10,133,201,169,0,133,202,32,141
9848 DATA196,169,10,133,201,169,0,133,202,32
     157,200,173,112,3,133,201,173,113,3
9868 DATA133,202,96
```

Listing 7: Revisions for 40 Columns

```
20 CR$=CHR$(13):DL$=CHR$(20):RB$="[RVS]
   [CL]":BL$="[38 SPACES]":DI$="
   [OFF] [CL]"
200 IFLEN(S$)   37THEN120
7010 X$=" ":IFRIGHT$(S$,1)="?"THENX$=STR$
     (SS)+"[RVS]"+LEFT$(BL$,34-LEN
     (STR$(SS)))
8520 PRINT"[YEL]"S$"[RVS]"LEFT$(BL$,38-LEN
     (S$)):NEXT:PRINT"[CH][CD]";:RETURN
```

Apple MICRO Calc Summary

Command Characters

@	Calculate
←	Delete previous character
RETURN	Move to end of next line
→	Move to end of last line
ESC	LOAD or SAVE screen
&	Clear screen lines

Characters Allowed

A...Z   0...9

/ * + − = . ( ) < > ? ↑

```
1  REM  *    MICROCALC
2  REM  *   BY P. DALEY
3  REM  * COPYRIGHT (C) 1982
4  REM  *   BY MICRO INK
5  NL=23
1Ø  HOME
2Ø CC = 95
3Ø  FOR II = 1 TO 38:BL$ = BL$ + " ": NEXT :BL$ = BL$ + "*"
4Ø  GOSUB 43Ø
5Ø  GOTO 75
6Ø  VTAB LL: PRINT S$(LL);
61  IF  MID$ (S$(LL),2,1) = "?" THE PRINT SS$(LL);
62  INVERSE
65  PRINT  CHR$ (CC); RIGHT$ (BL$, 38 -  LEN (S$(LL)) -  LEN
     (BB$(LL))): NORMAL
7Ø  RETURN
75 YY = Ø
8Ø LL = 1
9Ø  GOSUB 6Ø
1ØØ AA =  PEEK ( - 16384): IF AA <127 THEN 1ØØ
1Ø5  IF FLAG = 1 THEN FLAG = Ø: FOR II = 1 TO NL:SS$(II) = "": NEXT
11Ø  POKE  - 16368,Ø
12Ø AA = AA - 128
13Ø  IF AA = 64 THEN FLAG = 1: GOSUB 95Ø: GOTO 25Ø
131  IF AA = 38 THEN  GOSUB 53Ø:LL = 1: GOSUB 6Ø: GOTO 1ØØ
135  IF AA = 44 OR AA = 58 OR AA = 59 OR AA = 93 THEN 24Ø
14Ø  IF AA > 39 AND AA < 95 THEN 21Ø
15Ø CC = 32: GOSUB 6Ø
16Ø  IF AA = 13 THEN LL = LL + 2: IF LL > NL THEN LL = 1
17Ø  IF AA = 8 AND YY > 1 THEN S$ (LL) =  LEFT$ (S$(LL), LEN
     ( S$(LL)) - 1):YY = YY - 1: GOTO     18Ø
175  IF AA = 8 AND YY = 1 THEN S$ (LL) = "":YY = YY - 1
18Ø  IF AA = 21 THEN LL = LL - 2: IF LL < 1 THEN LL = NL19Ø  IF
     AA = 27 THEN  GOTO 1ØØØ
2ØØ  GOTO 23Ø
21Ø YY = YY + 1: IF YY > 38 THEN CC = 32: GOSUB 6Ø:LL = LL + 2:Y
     Y = Ø: GOTO 23Ø
22Ø S$(LL) = S$(LL) +  CHR$ (AA)
23Ø YY =  LEN (S$(LL)):CC = 95: GOSUB 6Ø
24Ø  GOTO 1ØØ
25Ø XY = LL
255  FOR II = 1 TO NL STEP 2
26Ø  IF  LEN (S$(II)) < 2 THEN 41Ø
27Ø  IF  MID$ (S$(II),2,1) < > "=" THEN 35Ø
28Ø  FOR JJ = 1 TO  LEN (S$(II))
29Ø  POKE 511 + JJ, ASC ( MID$ (S$ (II),JJ,1))
3ØØ  NEXT JJ
31Ø  POKE 511 + JJ,13
32Ø  CALL 768
35Ø  IF  MID$ (S$(II),2,1) < > "?" THEN 41Ø
36Ø  GOSUB 64Ø
39Ø SS$(II) =  STR$ (XX)
4ØØ LL = II:CC = 32: GOSUB 6Ø
41Ø  NEXT II
411 LL = XY:CC = 32: GOSUB 6Ø
412 LL = 1:CC = 95: GOSUB 6Ø
42Ø  GOTO 1ØØ
43Ø  FOR I = 1 TO 29:S$ = S$ + "*": NEXT
44Ø SS$ = "*                                    *"
45Ø  VTAB 5: PRINT S$: FOR I = 1 TO 1Ø
```

(continued)

(continued)

```
460 PRINT SS$: NEXT
470 PRINT S$: VTAB 8: HTAB 5: PRINT "MICRO CALC FOR APPLE"
480 VTAB 10: HTAB 5: PRINT "BY P. DALEY"
490 VTAB 12: HTAB 5: PRINT "COPYRIGHT (C) 1982"
500 DIM S$(25),SS$(25)
510 GOSUB 570
520 FOR I = 1 TO 500: NEXT
530 INVERSE : VTAB 1
535 HOME
540 FOR I = 1 TO NL STEP 2: PRINT BL$: IF I < > NL THEN  PRINT
542 S$(I) = ""
545 NEXT
550 NORMAL
560 RETURN
570 FOR I = 768 TO 802
571 READ A: POKE I,A: NEXT
572 DATA  165,184,72,165,185,72,
    169,0,133,184
573 DATA  169,2,133,185,32,89,213,169,0,133
574 DATA  184,169,2,133,185,32,70,218,104,133
575 DATA  185,104,133,184,96
630 RETURN
640 BB = ASC ( LEFT$ (S$(II),1))- 64
650 ON BB GOTO 670,680,690,700,710,720,730,740,750,760,770,780
   ,790,800,810,820,830,840,850,860,870,880,890,900,910,920
660 RETURN
670 XX = A: RETURN
680 XX = B: RETURN
690 XX = C: RETURN
700 XX = D: RETURN
710 XX = E: RETURN
720 XX = F: RETURN
730 XX = G: RETURN
740 XX = H: RETURN
750 XX = I: RETURN
760 XX = J: RETURN
770 XX = K: RETURN
780 XX = L: RETURN
790 XX = M: RETURN
800 XX = N: RETURN
810 XX = O: RETURN
820 XX = P: RETURN
830 XX = Q: RETURN
840 XX = R: RETURN
850 XX = S: RETURN
860 XX = T: RETURN
870 XX = U: RETURN
880 XX = V: RETURN
890 XX = W: RETURN
900 XX = X: RETURN
910 XX = Y: RETURN
920 XX = Z: RETURN
950 A=0:B=A:C=A:D=A:E=A:F=A:G=A: H=A:I=A:J=A:K=A:L=A:M=A:N=A: O=A
   :P=A:Q=A:R=A:S=A:T=A:U=A: V=A:W=A:X=A:Y=A:Z=A:RETURN
1000 HOME :D$ =  CHR$ (4)
1001 ONERR  GOTO 1000
1010 VTAB 10: INVERSE : PRINT "S";
1020 NORMAL : PRINT "AVE OR ";
1030 INVERSE : PRINT "L";
1040 NORMAL : PRINT "OAD?"
1050 PRINT : PRINT "<RETURN> FOR CATALOG."
1060 GET A$: PRINT : IF  ASC (A$) = 13 THEN  PRINT D$"CATALOG"
   : GET A$: GOTO 1000
1070 IF A$ = "S" THEN  GOSUB 1100
1080 IF A$ = "L" THEN  GOSUB 1200
1085 HOME : POKE 216,0
1090 CC = 32: FOR LL = 1 TO NL STEP 2 : GOSUB 60: NEXT :CC = 95:
   GOTO 80
1100 PRINT : PRINT "FILENAME?": INPUT A$
1110 PRINT D$"OPEN"A$
1120 PRINT D$"WRITE"A$
1130 FOR II = 1 TO 25
1140 PRINT S$(II)
1150 NEXT
1160 PRINT D$"CLOSE": RETURN
1200 PRINT : PRINT "FILENAME?": INPUT A$
1210 PRINT D$"OPEN"A$
1220 PRINT D$"READ "A$
1230 FOR II = 1 TO 25
1240 INPUT S$(II)
1250 NEXT
1260 PRINT D$"CLOSE": RETURN
```

MICRO

Lyco Computer Marketing & Consultants

TO ORDER CALL US **TOLL FREE 800-233-8760**

In PA 1-717-398-4079

SPECIAL PURCHASE

810 DISK DRIVE......$419.
NEC 8023 PRINTER... $449.
PERCOM AT-88 DISK... $395.

FREE DESIGNER TEE-SHIRT with PURCHASE of

800 48K RAM ... $489.00

ATARI®
A Warner Communications Company

ATARI HARDWARE

410 CASSETTE RECORDER ....$ 75.00
825 PRINTER....................$585.00
830 PHONE MODEM $149.00
850 INTERFACE.................$164.00

PACKAGES

CX482 EDUCATOR............. $119.00
CX483 PROGRAMMER......... $54.00
CX488 COMMUNICATOR.......$219.00
CX419 BOOKKEEPER $189.00
KX7104 ENTERTAINER......... $69.00

SOFTWARE

CXL4012 MISSILE COMMAND...$28.75
CXL4013 ASTEROID.............$28.75
CXL4020 CENTIPEDE $32.75
CXL4022 PACMAN $32.75
CXL4011 STAR RAIDER $34.75
CXL4004 BASKETBALL $26.75
CXL4006 SUPER BREAKOUT $28.75
CXL4008 SPACE INVADER.......$28.75
CX8130 CAVERNS OF MARS.....$31.75
CX4108 HANGMAN...............$12.75
CX4102 KINGDOM $12.75
CX4112 STATES & CAPITALS......................$12.75
CX4114 EUROPEAN COUNTRIES $12.75
CX4109 GRAPHIT..................$16.75
CX4121 ENERGY CZAR $12.75
CX4123 SCRAM...................$19.75
CX4101 PROGRAMMING I $19.75
CX4106 PROGRAMMING II.......$22.75
CX4117 PROGRAMMING III......$22.75
GLAXIAN $32.75
DEFENDER...................... $32.75
JUGGLES........................ $23.75
SPEED READING................ $55.75
CXL4007 MUSIC COMPOSER....$33.75
CXL4002 ATARI BASIC $45.75
CX8126 MICROSOFT BASIC..........................$65.75
CXL4003 ASSEMBLER EDITOR $45.75
CX8126 MACRO ASSEMBLER..................$69.75
CXL4018 PILOT HOME $65.75
CX405 PILOT EDUCATOR $99.75
CX415 HOME FILING MANAGER $41.75
CX414 BOOKKEEPER...........$119.75

NEW RELEASES

ATARI 1200 COMPUTER

THIRD PARTY SOFTWARE

EASTERN FRONT 1941 ....$25.50
OUTLAW/HOWITZER............$15.50
WIZARD of WAR $31.00
GORF.............................$31.00
FROGGER.........................$26.00
CHOP LIFTER.....................$27.75
APPLE PANIC.....................$23.75
PREPPIE $19.95
STAR WARRIOR...................$28.00
CRUSH,CRUMBLE,&CHOMP $23.00
SHOOTING GALLERY.............$19.95
VIDEO MATH FLASH $12.00
MY FIRST ALPHABET $25.50
BAHA BUGGIES $24.95
TEMPLE of ASPHAI...............$27.95
UPPER REACHES of ASPHAI $15.00
TRACK ATTACK...................$23.00
STAR BLAZER $25.00
LABYRINTH $23.00
SEA FOX $23.00
POOL 1.5..........................$26.95
SPEEDWAY BLAST (ROM).........$29.95

INHOME

400 KEY BOARD $99.75
PROTECTOR $24.95
NAUTILUS $24.95
SLIME $24.95
SUBMARINE COMMANDER (ROM) $36.95
JUMBO JET PILOT (ROM)...................$36.95
SOCCER (ROM) $36.95
KICKBACK (football ROM).........$36.95

PRINTERS

PROWRITER I $479.00
PROWRITER II .... $ CALL
OKIDATA 82A $419.00
OKIDATA 83A $639.00
OKIDATA 84 ....$1029.00
OKIDATA TRACTOR............$63.00
NEC 8023A $449.00
SMITH CORONA $589.00
STARWRITER............... $1475.00

BUSINESS SOFTWARE

ATARI WORD PROCESSING $109.00
LETTER PERFECT (ROM) $149.00
LETTER PERFECT (disc) $129.00
TEXT WIZZARD ....$ 89.00
DATA PERFECT ....$ 75.00
VISICALC $169.00
DATASAM/65 ....$125.00

JOYSTICKS

ATARI CX-40 ....$18.00
LESTICK $34.00
WICO COMMAND CONTROL.....$23.75
WICO RED BALL.................$26.75
WICO TRACK BALL...............$54.75
STICK STAND.....................$ 6.75

Book of ATARI Software (346 pages) .. $19.95

PERCOM

SINGLE DRIVE (SD) $399.00
SINGLE DRIVE (DD) $549.00
DUAL DRIVE (DD)................$869.00
DUAL HEAD (DD) $669.00

POLICY

In-Stock items shipped within 24 hours of order. Personal checks require four weeks clearance before shipping. No deposit for COD orders. PA residents add sales tax. All products subject to availability and price change. Advertised prices show 4% discount offered for cash. Add 4% for Mastercard and Visa.

TO ORDER
CALL TOLL FREE
800-233-8760
In PA 1-717-398-4079
or send order to
Lyco Computer
P.O. Box 5088
Jersey Shore, PA 17740

Digi-Draft for Atari 400/800

by Timothy Kilby

This drawing program allows you to construct images on an Atari GRAPHICS 7 screen. Convenient commands change colors, load and save, use alternate character sets, and draw points, lines, circles, and rectangles.

DIGI-DRAFT is joystick operated (port #1) and menu driven. Regular drawing is done in the default mode. You will have to press a key to select one of the options. You won't see an option in the menu to change all the playfield color registers; just press SELECT to initiate this option. The rest of the menu selections are

DIAGONAL LINE: Draws a diagonal line from the last plotted point.

RECTANGLE: Automatically draws a rectangle. Position cursor at one corner and press FIRE. Position cursor at opposite corner and press R.

CIRCLE: Draws arcs and circles, either filled or not. Circles near the screen's edge will continue to plot even if off screen.

BROKEN LINES: Draws hidden or dotted lines instead of solid lines.

FILL: Draws a line from the last plotted point, then fills all areas to the right with solid color. Fill lines cross screen unless stopped by a drawn line or plotted point.

TEXT: Controls text. Enter a text string, then position cursor. Text string will be plotted below and to the right of the cursor. Text will wrap around the edges of the screen. You may use all 256 characters including inverse and control characters.

LINECOLOR: Chooses color for lines; press 1, 2, or 3. Color 4 is the background color, so it acts as an eraser. Set specific hues with the SELECT option.

CLEAR: Erases the entire screen.

NEW FONT: Controls character style. Load a new character set from tape or disk. Characters from that set will appear only in the graphics window. Change character sets at any time.

SAVE SCREEN: Saves files. Press S to save your picture as a data file. If you are using a disk drive, enter a complete filename.

LOAD SCREEN: Loads files. Press L to load a previously saved data file.

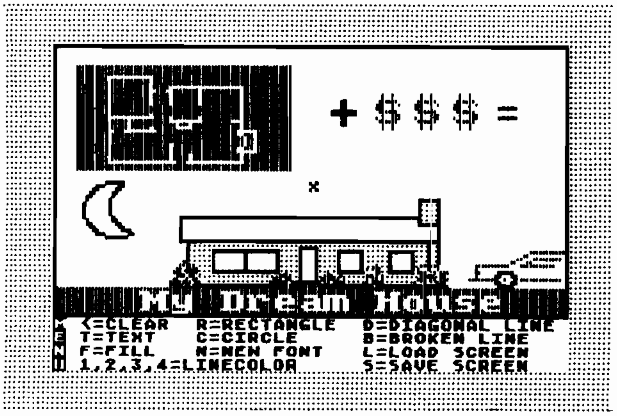
If you don't like DIGI-DRAFT's sound, simply eliminate lines 135, 420, and 1550. When you are saving or loading files to and from a disk, you must use complete filenames such as D1:FILENAME.EXT. Cassette users may wish to change nine lines to those in the cassette revision (listing 2): otherwise type "C:" whenever you're asked to enter a filename. Typing the remark statements is not necessary. Type all underlined spaces and characters in inverse by pressing the Atari logo key. Lines 8030 and 9050-9070 must be typed exactly as printed or the program will crash. You can save memory by changing frequently used constants like 0 and 1 to variables (but this will make reading the code more difficult).

DIGI-DRAFT's Design

DIGI-DRAFT's design begins with a simple GRAPHICS 7 drawing program, then uses advanced program design for unique results. The cursor has been replaced with a cursor-like player. You have seen players in the form of spaceships, aliens, or six-shooting cowboys. DIGI-DRAFT's player is an X character that marks the pen's position. This player is independent of all other graphics; it is able to dance around the screen without erasing lines. Points will be plotted and lines will be drawn based on the player's position on the screen.

Another special feature is a text-plotting routine that will draw text characters in the graphics window. It permits you to use all 256 characters, including inverse video and control characters. If a custom character set is loaded — a nice way to add architectural, electronic, game-piece, or other special symbols — those characters can be used in the graphics window. Furthermore, you can load new character sets at any time so that characters from several different sets could be used within one design. The routine plots points rather than prints characters, so any of four colors could be used. Special effects such as 3-D shadowed text and color-on-color are easily implemented. Atari characters are the default

Figure 1: Sample Screen from DIGI-DRAFT



characters; if you want custom characters you will have to create them yourself with a character editor program. There are several excellent character set editors available commercially that will assist you in creating character data files.

Whenever a custom character set is loaded, the character base shadow register at $2F4 is set to point to the new character set. The new characters would be used anywhere on the screen, including the text window, if it were not for a display list interrupt (DLI) routine added to DIGI-DRAFT. The DLI routine is a short machine-language routine that makes the computer display only Atari characters in the text window and adds extra color as a bonus.

Just before ANTIC instructs the CTIA (or GTIA) chip to display the text window at the bottom of the screen, once every sixtieth of a second, the DLI routine is called up to switch to Atari characters and change two colors. Later, as the screen begins plotting anew, all the shadow register data for graphics window colors and the custom character set location are loaded into the appropriate registers. Thus, the computer continually switches between given values for the text window and your colors and characters in the graphics window. Line 9050 of listing 1 contains the numerical data that create this routine. In assembly language it looks like this:

```
PHA
TXA
PHA
TYA
PHA
LDA #$E0   ;use Atari character set
LDX #$2C   ;light yellow background (or color of
            your choice)
LDY #$06   ;dark text color
STA $D40A  ;wait for WSYNC
STA $D409  ;store Atari character pointer in
            CHBASE
STX $D018  ;store background color in COLPF2
STY $D017  ;store text color in COLPF1
PLA
TAY
PLA
TAX
PLA
RTI
```

The DLI routine, along with the machine-language player movement and input/output routines, is located in program memory, thereby freeing page six for screen dumps or other machine-language routines.

Your drawings can be saved by DIGI-DRAFT and reloaded at a later time. The screen-saving subroutine stores all graphics data and the colors from the color shadow registers into a data file. The routine saves data for the graphics window directly from screen memory, instead of retrieving pixel data from the screen — a rather slow procedure. The screen-loading subroutine does just the opposite. The loading subroutine, beginning at line 4000, could be used with other programs to load your picture data files to a GRAPHICS 7 screen. Thus your screen designs could be incorporated in a variety of program applications. Just imagine an electronics program with schematic drawings selected by menu, or an adventure game with code books or detailed maps.

***Listing 1:* DIGI-DRAFT Program Listing**

```
0 REM DIGI-DRAFT   (C) 1982 TIM KILBY
5 OVERSCAN=0.8:REM OVERSCAN DETERMINES PERFECT CIRCLES.  CHANGE TO VALUE FROM 0.
50 TO 2.00 FOR OVAL SHAPE.
10 GOTO 7010
20 REM  READ JOYSTICK
30 A=STICK(0):B=STRIG(0):I=PMB+Y-1:IF A=11 AND X>44 THEN X=X-1:POKE PL,X:RETURN
40 IF A=7 AND X<203 THEN X=X+1:POKE PL,X:RETURN
50 IF A=13 AND Y<94 THEN D=USR(XD,I):Y=Y+1:RETURN
60 IF A=14 AND Y>15 THEM D=USR(XU,I):Y=Y-1:RETURN
70 IF A=9 AND X>44 AND Y<94 THEN X=X-1:POKE PL,X:D=USR(XD,I):Y=Y+1:RETURN
80 IF A=5 AND X<203 AND Y<94 THEN X=X+1:POKE PL,X:D=USR(XD,I):Y=Y+1:RETURN
90 IF A=10 AND X>44 AND Y>15 THEN X=X-1:POKE PL,X:D=USR(XU,I):Y=Y-1:RETURN
100 IF A=6 AND X<203 AND Y>15 THEN X=X+1:POKE PL,X:D=USR(XU,I):Y=Y-1
110 RETURN
120 REM  DRAWING ENTRY POINY
130 GOSUB JOY:IF B=0 THEN COLOR C:GOSUB 410
135 SOUND 0,0,0,0
140 IF PEEK(K)<>RET THEN GOSUB 250
150 IF PEEK(53279)=5 THEN ? C$;B$:GOSUB 2010:GOSUB MENU
160 GDTO 130
170 REM  LINE COLOR CHANGE
180 IF KEY=49 THEN C=1:GOSUB 230
190 IF KEY=50 THEN C=2:GOSUB 230
200 IF KEY=51 THEN C=3:GOSUB 230
210 IF KEY=52 THEN C=0:GOSUB 230
220 RETURN
230 FOR I=0 TO 20:POKE 704,0:POKE 704,14:NEXT I:POKE 704,88:RETURN
240 REM  READ KEYBOARD
250 GET #4,KEY:IF KEY=78 THEN ? C$:GOSUB 5010:GOSUB MENU
260 GOSUB 180:IF KEY=68 THEN GOSUB 510
270 IF KEY=60 THEN 9000
280 IF KEY=67 THEN ? C$:GOSUB 910
290 IF KEY=84 THEN ? C$:GOSUB 1210:POKE 702,64:POKE 694,0
300 IF KEY=70 TNEN GOSUB 810
310 IF KEY=82 THEN GOSUB 710
320 IF KEY=76 THEN ? C$:GOSUB 4010:GOSUB MENU
330 IF KEY=66 THEN ? C$:GOSUB 1510
340 IF KEY=83 THEN ? C$:GOSUB 3010:GOSUB MENU
350 RETURN
360 REM  MENU
370 POKE 752,1:POKE 82,0
380 ? C$;"M <=CLEAR  R=RECTANGLE  D=DIAGONAL LINE E T=TEXT   C=CIRCLE     B=BROK
EN LINE"
390 ? "N F=FILL   N=NEW FONY   L=LOAD SCREEN   U 1,2,3,4=LINECOLOR     S=SAVE SC
REEN";:POKE 82,2:RETURN
400 REM  DRAW
410 X1=X-44:Y1=Y-15:PLOT X1,Y1:X2=X1:Y2=Y1
420 SOUND 0,X1+(79-X1)*(X1>78)*2+T1+(39-Y1)*(Y1>38)*2+20,10,4
430 RETURN
500 REM  DIAGONAL
510 POKE 53279,0:X1=X-44:Y1=Y-15:COLOR C:DRAWTO X1,Y1:RETURN
700 REM  RECTANGLE
710 COLOR C:X1=X-44:Y1=Y-15:DRAWTO X2,Y1:DRAWTO X1,Y1:DRAWTO X1,Y2:DRAWTO X2,Y2:
PLOT X1,Y1:X2=X1:Y2=Y1:RETURN
800 REM  FILL
810 X1=X-44:Y1=Y-15:POSITION X1,Y1:POKE 765,C:XIO 18,#6,0,0,"S:":RETURN
900 REM  CIRCLES
910 F=0:? "Do you want your circle FILLED (Y/N)?";:GET #4,KEY:IF KEY=89 THEN F=1
930 ? C$;D$;"Position cursor at one point on EDGE  of circle or arc and press FI
RE."
940 GOSUB JOY:IF B=1 THEN 940
950 POKE 53279,0:FOR D=1 TO 50:NEXT D:X1=X-44:Y1=Y-15
960 ? C$;D$;"Now position the cursor at the CENTER of your circle or arc and pre
ss FIRE."
970 GOSUB JOY:IF B=1 YHEN 970
980 POKE 53279,0:C1=X-44:C2=Y-15:R=INT(SQR(ABS(C1-X1)^2+ABS(C2-Y1)^2)+0.5):? C$;
990 ? "   - Change color by pressing -               1 ,  2 ,  3 ,or  4 "
995 ? :? "     Press  SPACE BAR  to exit";
```

(continued)

Listing 1 *(continued)*

```
1000 FOR N=270 TO 629:X2=C1+R*COS(N):Y2=C2+R*OVERSCAN*SIN(N):IF PEEK(K)<>RET THE
N GOSUB 1050
1020 IF X2<1 OR X2>159 OR Y2<1 OR Y2>79 THEN NEXT N
1030 COLOR C:PLOT X2,Y2:IF F=1 THEN DRAWTO C1,C2
1040 NEXT N:GOTO 1000
1050 GET #4,KEY:IF KEY>48 AND KEY<53 THEN GOSUB 180
1060 IF KEY=32 THEN C=1:POP :GOSUB MENU:RETURN
1070 RETURN
1200 REM  TEXT
1210 POKE 752,0:? "Enter TEXT string you wish printed.":INPUT B$:IF B$="" THEN G
OSUB MENU:RETURN
1220 POKE 752,1:? C$;D$;"Now position cursor at beginning      location and pres
s FIRE button."
1230 GOSUB JOY:IF B=1 THEN 1230
1250 X1=(X-44):Y1=(Y-15)
1260 FOR A=1 TO LEN(B$):F=0:IF ASC(B$(A,A))>127 THEN F=RET
1270 IF X1>152 THEN X1=0:Y1=Y1+8
1280 IF Y1>72 THEN Y1=0
1290 B=ASC(B$(A,A)):GOSUB 1390:R=PEEK(756)*256+B*8:FOR I=0 TO 7:COLOR C:D=ABS(PE
EK(R+I)-F)
1300 IF D>127 THEN D=D-128:PLOT X1,Y1+I
1310 IF D>63 THEN D=D-64:PLOT X1+1,Y1+I
1320 IF D>31 THEN D=D-32:PLOT X1+2,Y1+I
1330 IF D>15 THEN D=D-16:PLOT X1+3,Y1+I
1340 IF D>7 THEN D=D-8:PLOT X1+4,Y1+I
1350 IF D>3 THEN D=D-4:PLOT X1+5,Y1+I
1360 IF D>1 THEN D=D-2:PLOT X1+6,Y1+I
1370 IF D>0 THEN PLOT X1+7,Y1+I
1380 NEXT I:X1=X1+8:NEXT A:GOSUB MENU:B$=CHR$(253):RETURN
1390 IF B>127 THEN B=B-128
1400 IF B>31 AND B<96 THEN B=B-32:RETURN
1410 IF B<32 THEN B=B+64
1420 RETURN
1500 REM  BROKEN LINES
1510 ? "          BROKEN LINES MODE":? :? "    - Press  SPACE BAR  to exit -";
1520 COLOR C:IF A<>11 AND A<>7 AND Y1/4=INT(Y1/4) THEN COLOR 0
1530 IF A<>14 AND A<>13 AND X1/5=INT(X1/5) THEN COLOR 0
1540 GOSUB JOY:IF B=0 THEN GOSUB 410
1550 SOUND 0,0,0,0
1560 IF PEEK(K)<>RET THEN 1580
1570 GOTO 1520
1580 GET #4,KEY:IF KEY=32 THEN GOSUB MENU:RETURN
1590 IF KEY>48 AND KEY<53 THEN GOSUB 180
1600 GOTO 1520
2000 REM  COLOR CHANGE
2010 ? "Use joystick to change four colors.   Press FIRE also to change intensit
y."
2020 ? :? "     - Press any key to exit -";
2030 A=STICK(0):B=STRIG(0):IF PEEK(K)<>RET THEN RETURN
2040 POKE 708,C0:IF A=14 AND B<>0 THEN C0=C0+16:IF C0>RET THEN C0=C0-256
2050 POKE 709,C1:IF A=7 AND B<>0 THEN C1=C1+16:IF C1>RET THEN C1=C1-256
2060 POKE 710,C2:IF A=13 AND B<>0 THEN C2=C2+16:IF C2>RET THEN C2=C2-256
2070 POKE 712,C4:IF A=11 AND B<>0 THEN C4=C4+16:IF C4>RET THEN C4=C4-256
2080 IF B=0 AND A=14 THEN C0=C0+2:IF C0>RET THEN C0=C0-256
2090 IF B=0 AND A=7 THEN C1=C1+2:IF C1>RET THEN C1=C1-256
2100 IF B=0 AND A=13 THEN C2=C2+2:IF C2>RET THEN C2=C2-256
2110 IF B=0 AND A=11 THEN C4=C4+2:IF C4>RET THEN C4=C4-256
2120 GOTO 2030
3000 REM  SAVE SCREEN
3010 POKE 752,0:? "Enter filename for drawing.":INPUT F$:POKE 752,1:? C$:IF F$="
" THEN RETURN
3020 TRAP 3080:POKE PL,0:CLOSE #1:OPEN #1,8,0,F$
3030 FOR I=708 TO 712:PUT #1,PEEK(I):NEXT I
3040 POKE 852,PEEK(88):POKE 853,PEEK(89):POKE 856,128:POKE 857,12:POKE 850,11
3050 J=USR(ADR(CIO$))
3060 POKE 54286,192:? C$;D$;"      Screen has been saved as:":? "          ";F
$;B$:FOR D=1 TO 500:NEXT D
3070 POKE PL,X:TRAP 130:RETURN
3080 POKE 54286,192:? C$;D$;"Sorry! Check connections - try again.";B$:FOR D=1 T
O 400:NEXT D:GOTO 3070
4000 REM  LOAD SCREEN
4010 POKE 752,0:? "Enter the drawing's filename.":INPUT F$:POKE 752,1:? C$:IF F$
="" THEN RETURN
4020 POKE PL,0:TRAP 4080:CLOSE #1:OPEN #1,4,0,F$
4030 FOR I=708 TO 712:GET #1,C:POKE I,C:NEXT I
4040 POKE 852,PEEK(88):POKE 853,PEEK(89):POKE 856,128:POKE 857,12:POKE 850,7
4050 J=USR(ADR(CIO$))
4060 POKE 54286,192:C=1
4070 POKE PL,X:TRAP 130:C0=PEEK(708):C1=PEEK(709):C2=PEEK(710):C4=PEEK(712):RETU
RN
4080 POKE 54286,192:? C$;D$;"Sorry!  Problems with that file.     Try again.";B
$:FOR D=1 TO 300:NEXT D:GOTO 4070
5000 REM  LOAD CHARACTER SET
5010 POKE 752,0:? "Enter filename for your character set.(Press RETURN for ATARI
 characters.)  ";
5020 TRAP 5070:INPUT F$:POKE 752,1:IF F$="" THEN ? B$:POKE 756,224:T=0:RETURN
5030 CLOSE #1:OPEN #1,4,0,F$:POKE 852,0:POKE 853,CS:POKE 856,0:POKE 857,4:POKE 8
50,7
5040 J=USR(ADR(CIO$))

5050 POKE 756,CS:POKE 54286,192:? C$;D$;"Load complete . . . . .":FOR D=1 TO 250
:NEXT D:T=1
5060 TRAP 130:RETURN
5070 POKE 54286,192:? C$;D$;"Problems with that file. Try again.";B$:FOR D=1 TO
300:NEXT D:GOTO 5060
7000 REM  TITLE SCREEN
7010 GRAPHICS 17:POKE 708,214:POSITION 5,7:? #6;"DIGI-DRAFT":POSITION 4,10:? #6;
"BY TIM KILEY"
8000 REM  INITIALIZATION
8010 DIM F$(15),C$(1),B$(19),D$(1),ML$(71),CIO$(6)
8020 FOR D=1 TO 6:READ A:CIO$(D,D)=CHR$(A):NEXT D
8030 DATA 104,162,16,76,86,228
8040 C0=150:C1=70:C2=220:C4=14:C=1:RET=255:T=0:K=764:JOY=30:PL=53248:MENU=370:DE
G :TRAP 130
8050 B$=CHR$(253):C$=CHR$(125):D$=CHR$(29):N=ADR(ML$):XU=N+29:XD=N+50
8060 POKE 512,N-INT(N/256)*256:POKE 513,INT(N/256)
8070 FOR I=0 TO 70:READ A:POKE N+I,A:NEXT I
8080 OPEN #4,4,0,"K:"
8090 I=PEEK(106)-16:POKE 106,I:CS=I+12
9000 GRAPHICS 7:POKE 708,C0:POKE 709,C1:POKE 710,C2:POKE 712,C4:I=PEEK(560)+256*
PEEK(561):POKE I+83,141
9010 POKE 54286,192:X=123:Y=55:A=PEEK(106)-2:POKE 54279,A:PMB=256*A:POKE 559,46:
POKE 53277,3:POKE PL,X
9020 IF PEEK(PMB)<>0 THEN FOR I=PMB TO PMB+128:POKE I,0:NEXT I
9030 RESTORE 9080:POKE 704,88:FOR I=PMB+Y TO PMB+2+Y:READ A:POKE I,A:NEXT I
9040 GOSUB MENU:POKE 756,CS*(T=1)+224*(T=0):GOTO 130
9050 DATA 72,138,72,152,72,169,224,162,44,160,6,141,10,212,141,9,212,142,24,208,
140,23,208,104,168,104,170,104,64
9060 DATA 104,104,133,178,104,133,177,160,1,177,177,136,145,177,200,200,192,5,20
8,245,96
9070 DATA 104,104,133,178,104,133,177,160,4,177,177,200,145,177,136,136,192,255,
208,245,96
9080 DATA 20,8,20
```

Listing 2: Cassette Revision to DIGI-DRAFT

```
3010 ? "Are you sure?  (Y/N) ";:GET #4,KEY:IF KEY<>89 THEN RETURN
3015 ? C$;D$;"Insert a blank cassette, press RECORD and PLAY, then press RETURN.
":F$="C:"
4010 ? "Are you sure?  (Y/N) ";:GET #4,KEY:IF KEY<>89 THEN RETURN
4015 ? C$;D$;"Insert screen data tape, press PLAY,  then press RETURN.":F$="C:"
5010 ? "Press  A  for ATARI characters or  N  to load new character set.   ";:GE
T #4,KEY
5012 IF KEY=65 THEN ? B$:POKE 756,224:POKE 752,1:T=0:RETURN
5014 IF KEY<>78 THEN RETURN
5016 ? C$;D$;"Insert character data tape, press    PLAY, then press RETURN.";B$
:F$="C:"
5020 IF PEEK(K)<>12 THEN 5020
```

MICRO

MICRObits

AIM 65 Real Time Clock

Provides hour, minute, second, day of week, day, month, year. Twelve- or 24-hour format. Pin-compatible with AIM expansion connector (also SYM, KIM). Four switch-selectable interrupts. Nicad battery backup. Industrial quality board 4.5 × 6. All ICs socketed. Single 5V supply. Twenty-two-page manual. All software included. Bare board $29.00. Complete A&T $93.00, includes batteries. Add $4.00 shipping and handling. CA residents add 6%.

Data Design Group
P.O. Box 3318
La Jolla, CA 92038
(619) 265-6940

Lessons in Algebra

An easy and fun way to learn the basic elements of high school algebra. Apple computer diskette $29.95. 30-day money-back guarantee if not satisfied.

George Earl
1302 So. General McMullen Dr.
San Antonio, TX 78237

Target-an AIM 65 Newsletter

Need information for your AIM 65 computer? News, software, and hardware are examples of items covered in the newsletter. Yearly subscription rates are $7.00 in the US and Canada, $12.00 elsewhere. Back issues are available beginning with 1979 at the same per year rate.

Target
c/o Donald Clem
RR#2
Spencerville, OH 45887

The State of the Art in Astro-Software

Wide range of astrological and astronomical software of the highest quality. From powerful (and income-producing) astrological charting service packages and printing interpretation packages, to super-accurate computer ephemerises. For all Commodore computers, Apple II Plus, and TRS-80.

Matrix Software
315 Marion Avenue
Big Rapids, MI 49307

MICRO

It Pays to Write for MICRO

Get paid for your ideas: write for MICRO! Thousands of people read MICRO every month. MICRO is sold in computer stores and on newsstands worldwide. Send for a copy of our Writer's Guide now. Our author payment rate is competitive with the leading magazines in the industry.

We welcome articles on any aspect of 6502/6809/68000 hardware and software for the Apple, Atari, CBM/PET, TRS-80 Color Computer, VIC, OSI, 6809, or 68000.

1983 Features:

April — Communications
May — Wave of New Computers
June — Operating Systems
July — Hardware
August — Word Processing
September — Education
October — Programming Techniques
November — Games
December — New Microprocessors

The Computer Revolution Reaches the Community

by Emmalyn H. Bentley

Many questions confronting the public today concern the computer revolution's role in our daily lives. How is this revolution affecting business, education, communication, recreation, and even the drudgery of routine tasks we all must perform? Can the field of computer science be assimilated easily into our day-to-day existence if we are not conversant with high technology? What practical steps can we take to prevent a "future-shock" impact on our sensibilities?

Some of these questions are being resolved at the newly created Microcomputer Learning Center in Nashua, NH. The project is a combined effort of the Arts and Science Center, Nashua and The Computer Mart of New Hampshire, Inc., and has education at the heart of its intent. It is a merging of two entirely different kinds of organizations (a non-profit institution and a profit-making business) to create a unique blend of expertise and resources. The result is an exciting learning center complete with knowledgeable educators, hardware and software, and an extensive library of computer literature.

Presently the Learning Center has ten Apple II Plus computers with disk drives and video monitors, and three printers. A resource center has been established to provide the latest in applications software, books, and journals. Hands-on instruction is offered in such courses as Computer Literacy, VisiCalc, Word Processing, and BASIC, to name only a few. In-depth workshops cover LOGO and PILOT, computer graphics, specific languages, software evaluation of computer-assisted instruction, and administrative management and computing for remediation. Although the Center uses only Apple computers, this is not a factor when decisions are made about what courses to take. The Learning Center teaches applications and word processing that are transferrable and generalized; it offers across-the-board learning. *Knowledge* is what it is all about.

According to Stuart Carduner, NH Children's Museum director and member of the Learning Center's Task Force, the long-range goal is to have a central resource center that provides the community with access to many types of computers, hardware, software, and information to enhance the learning process, as well as the competency to pass on this knowledge to the public in an informative and educated fashion. The Center will go "wherever there is a need — the limit is only what the community wants," explains Mr. Carduner. Plans include adding more specific courses of study as the demand increases — advanced LOGO for teachers, supplemental extensions to the VisiCalc family, and Pascal. But the core of introductory programs will continue to be a very integral part of the learning system.

Because the center is community-oriented, the general public is encouraged to particpate in this inspiring educational venture. Businesses can send employees to specially custom-designed training programs. (One Nashua school sent four secretaries to the center to learn word processing.) Teachers are encouraged to actively participate in such workshops as Computer Applications in the Classroom, Introduction to LOGO, and Computer Literacy for Teachers. To help the schools prepare for a future in which computer courses may be required, teachers are invited to bring their classes to the Learning Center for a working introduction to computers designed to fit any age group. Individuals are offered one-day sessions to help balance checking accounts and budgets, perform diet analysis, or even learn a new computer game. Intensive courses can help the novice grasp the elements of computer savvy or provide added job skills.

"The Children Know the Most"

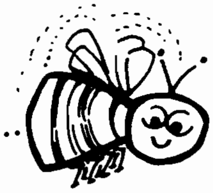
The Center is attracting a cross-section of society — young and old, with little or no knowledge of computing. According to Gerry Paquin, M.Ed., Marketing Director and Education Consultant for The Computer Mart of NH and task force member of the Learning Center, of all the people enrolling in the classes "the children know the most!" They are being exposed to computers at school and at home, or have had some prior experience on the few terminals the NH Children's Museum (at the Arts and Science Center) provided before the Learning Center opened. The adults are the *real* novices; they are curious to know "How can it (the computer) help organize my information, whether it be at home or at work? How can it simplify certain things that I do repetitively?" There is a drive for learning, a gusto for "I've got to find out about this because it's new and it's happening and I want to be in on it." Mr. Paquin claims "The hardest part about computers is to get people to sit down and try it and to deal with their preconceptions about what it is." Eight or nine of ten class members have never used a computer before. They don't know what a computer can do and, in most cases, are truly amazed. "Wow! A computer can do that?" is a familiar refrain.

The task force members at the Learning Center are actively seeking people from a variety of backgrounds — the businessman, educator, student, and homemaker. Recently they offered a course to the Nashua Women's Club. Mr. Paquin tells of one instance where a computer course, taught in a nursing home, was an incredible success. Courses for children are provided as an extension to what they have learned at school, filling a need to give the youngsters a place to further their computer education. One of the most popular courses that "fills like crazy" is Advanced BASIC for Young People. Mary Gasiorowski (task force member) talks about a mailing-list program she taught, in which one youngster wrote an imaginative title page that moved an envelope across the page to a mailbox!

The Future is Now

One final comment about the revolutionary growth of computers in the community. Three or four years ago the computer world was an entity of its own and appeared to consist of two groups: the ultra-serious, high-technology computerist who sat in his lonely corner preparing incomprehensible programs for inexplicable applications; and the computer hobbyists who gathered on Saturday mornings in computer shops to talk, talk, talk! (Mr. Paquin believes one of the positive things to come out of this is the willingness of customers and shop owners to exchange information and ideas — a phenomenon that does not occur in any other phase of business.) These preconceived stereotypes no longer exist. The staid programmer has emerged from his dark recess with a sense of altruistic values that he now shares with others; there is a comraderie, a feeling of "we're doing this because it's fun." Apple, for example is a "potential movement" with its festivals and clubs, and this enthusiasm is spreading. It is an exciting age when the most ordinary of us have an inclination to become computer-literate, to get on with the business of living in this inescapably technical world in a more meaningful fashion. The Microcomputer Learning Center in Nashua is helping to make this happen. By bringing the micro to the general public, people are able to get in touch with one another in the computer environment in a very human kind of way. MICRO

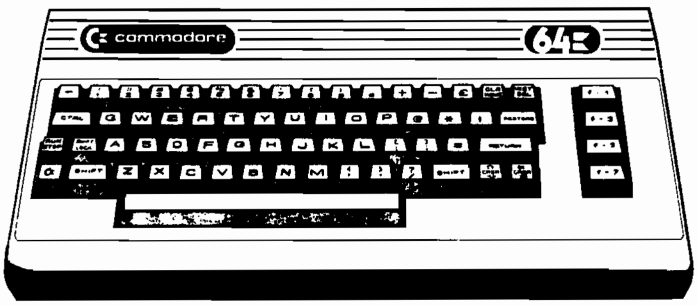
BUSICALC

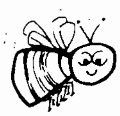


BUSICALC A Honey of an Electronic Spreadsheet

Why electronic spreadsheet programs?

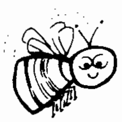
Electronic spreadsheet programs allow the user to create a gridsheet, spreadsheet, worksheet, or any other table of information, using the memory of the computer as pencil and paper. The computer display or terminal acts as a window through which the user views the information as it is entered. Textual information (such as headings), numerical values, and formulas can easily be entered into the spreadsheet.





For Commodore 64
For Commodore VIC 20
For Commodore PET/CBM 40 columns
For Commodore CBM 80 column/SuperPet

BUSICALC Your Computer Drone for Repetitive Calculations



The outstanding advantage of using a computer is that it acts not only as a pencil and paper but as a perfect eraser and an automatic calculator. The user can quickly and easily make any number of alterations to the data within the table. The BUSICALC will evaluate any formula using the data that has been entered. Further, it retains the formulas and displays the resulting value. With BUSICALC controlling the entry of data, providing a comprehensive memory, and performing arithmetic, the preparation of a spreadsheet is faster and more accurate than if it were prepared by hand.

BUSICALC With the Sting Removed from the Prices

BUSICALC 20.................................... only **$49.00** for the VIC 20
BUSICALC 64.................................... only **$69.00** for the CBM 64
BUSICALC 40.................... only **$79.00** for the original 40 column PET/CBM
BUSICALC 80......... only **$89.00** for the original 80 column CBMs and SuperPets

BUSICALC AVAILABLE NOW FROM YOUR LOCAL DEALER
(800) 227-9998
FOR THE NAME OF YOUR NEAREST DEALER

California, Canada, Alaska and Hawaii please call (415) 965-1735

Skyles Electric Works
231G South Whisman Road
Mountain View, CA 94041

Europe please contact Supersoft, Winchester House, Canning Road, Harrow Wealdstone, Middlesex, England HA3 7SJ, Tel. 01 861 1166

Circle No. 38

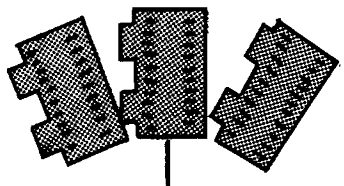
WE'VE BEEN BUSY!

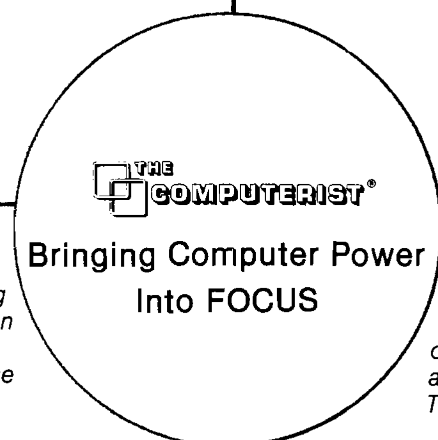
HAVEN'T HEARD FROM US IN A WHILE

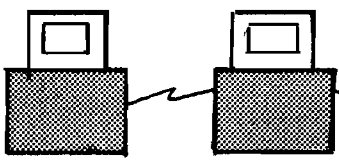
WONDER WHAT WE'VE BEEN DOING?

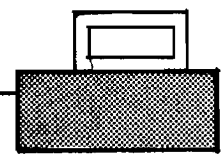
A LOT!

We have been upgrading our board level products—
If you are expanding your existing AIM, SYM or KIM computer,
or building a system from the ground up,
check out these updated boards—
DRAM PLUS now 16/32/64/80/128 K bytes
plus up to 8K static RAM or
EPROM Programmer, VIA's...
VIDEO PLUS/MICRO Plus has been refined...
FLEXI PLUS has gone to 64K dynamic RAM,
DMA for the Floppy and IEEE 488, 2 MHz...





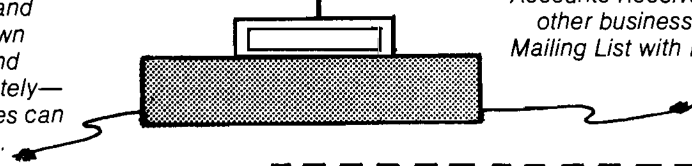




We have developed a complete Typesetting System which expands the capabilities of an EditWriter to accept data from the phone lines, produce plain paper proofing copy, use improved word processing techniques, plus run a variety of printing/publishing oriented packages.

We have given "SERVICE CALL" a new meaning. Using our Master/Slave programs, a system can be examined, tested and often fixed via the telephone!!! Blown data disks, strange malfunctions, and other 'quirks' can be cured immediately—over the telephone. Software updates can also be sent directly to your system.

We have been expanding the capabilities of the FOCUS™ System—and have added a lot of software:
TypePlus — a complete word processing system
SpellingPlus — interactive spelling checker
Dynacalc™ — the latest in VisaCalc™ type Spread Sheets
Accounts Receivable/Accounts Payable and other business packages
Mailing List with Form Letters and more...



YES — we have been very busy. In addition to our development work in both hardware and software, we have been providing board level products to a variety of OEMs and end users.* We don't have time right now to tell you about all of the things we can provide, but, write or give us a call and we'll make time!!!

Call us at 617/256-3649 or call our FOCUS System at 617/250-1460 (300 BAUD)

**Significant OEM Discounts Start At Only Five (5) Units.*
FOCUS, DRAM PLUS, VIDEO PLUS, MICRO Plus, FLEXI PLUS, and TypePlus Trademark TCI
Dynacalc TM Computer Systems Center
VisaCalc TM VisaCorp

Circle No. 39

SEND ME MORE INFORMATION

Name ____________________

Company ____________________

Address ____________________

City ____________ State ______ Zip ________

MY INTEREST IS: ____________________

____

THE COMPUTERIST
34 Chelmsford St., Chelmsford, MA 01824
617-256-3649

BANNER: A Display Program for the Color Computer

by Bryan Christiansen

If you have ever wanted to see your name in lights, I have just the program for you. The routine is written for the TRS-80 Color Computer and displays large-size messages on the CRT screen. In addition, I have included some powerful machine-language loading techniques that provide efficient and safe handling of machine-language subroutines.

My program lets you input any message with capital letters and punctuation. You can vary the height, width, and speed of the display. The program prints the large letters and the background on the video monitor with any character the computer can generate in the alphanumeric mode. The message scrolls from left to right, just like on Wall Street!

The program is smooth and versatile because machine-language subroutines do a lot of the work. One routine fills the screen with any character; another routine does the printing and scrolling.

Here is a brief explanation of how my program works. First, all the character definitions are in the form of string arrays. From those strings in BASIC, the program makes a table in memory, which the machine-language subroutine reads. There is also an array that contains the information required to construct each letter. The BASIC program points to the memory location of strings to be printed, and the machine-language subroutine takes care of the rest. The BASIC program also controls the speed, the size, and the colors of the letters.

This article includes information on how you can combine the BASIC code with the machine-language routines. You can use this method with your own programs. First, the theory is to take a BASIC program and tack a machine-language program to the end of it. BASIC has two memory locations that hold the address of the end of a BASIC program in memory. That address is used for saving BASIC programs to tape. Also, when you edit anything in BASIC, the address is used to find out how much memory has to be moved. Since BASIC does not use the address for listing or running a program, you do not have to worry about your machine-language program interferring with your BASIC program. While it is reading a program, BASIC looks for three zeros to determine the end of program. All you have to do is find out the address of the end of your BASIC program and add the length of your machine-language program to it. Then change the end-of-program memory location to the number you calculated.

Finally, store your machine-language program in memory and you have combined the two programs. The machine-language program moves up and down in memory with your BASIC program so you do not have to worry about changing your BASIC program. Also, since there is no need to reserve memory for machine-language routines, you are using the memory in your computer economically. Accidentally clearing string space will not destroy the machine-language routines.

There are some limitations to the technique: the machine-language routine has to be position-independent, and since the routine moves every time you change your BASIC program, you may have trouble finding your machine-language program. BASIC can find it easily just by PEEKing the address those two memory locations hold. Since your machine-language program does not change its length, BASIC only has to subtract to find the beginning of your machine-language program. You do it like this: enter

```
PRINT PEEK (27)*256 + PEEK(28)
```

Copy down the address the computer gives, add the length of your machine-language subroutine to the number you have on paper, then POKE the new number back into those memory locations. Since the number will be different for different programs, I have put a # sign where you are to insert the number you calculated. Enter

```
POKE 27,INT(#/256) : POKE 28, # - INT(#/256)*256
```

Take the first number you wrote down and add one to it. You can start POKEing in your machine-language program at that memory location. Now save your BASIC program. The next time you CLOAD, it will load in your BASIC program and your machine-language program at the same time. To find the beginning of your machine-language program simply add a line like this to your BASIC program (put the length of your machine-language program where I have a # sign):

A = PEEK (27)*256 + PEEK (28) − #

To combine the machine language with the BASIC in my program start by entering this line:

30 EN = PEEK (27)*256 + PEEK (28) − 972

Then enter

PRINT PEEK (27)*256 + PEEK (28)

Write down the number and add 972 (routine length) to the number. Put that number where I have a # sign and enter

POKE 27,INT(#/256) : POKE 28, # − INT(#/256)*256

Run the program. When it starts scrolling, press the BREAK key and delete lines 34 to 90. Finally, enter

1290 GOTO 1430

Now the program is finished. You can save a copy and when you load it in the next time it will load in the machine language also. Since the program gets rid of the wait at the beginning, the program starts scrolling immediately and takes less memory.

Here is a simple outline of the program.

10-30	SET UP
40-100	M.L. LOADER
110-1270	STRING PATTERNS
1280-1290	MORE SET UP
1300-1420	CHARACTER SET LOADER
1430-1570	MAIN LOOP
1580-1670	MENU
1680-2020	ROUTINES FOR CHANGING PARAMETERS

You can change the starting message by changing line 30. To change the startup parameters change line 1280. All the REM statements are between the ten intervals so you dc not have to type them. I suggest you save the program to tape before running it because a mistake might destroy the program. When you have finished and you want to be sure all the letters are right, enter nothing for a message. The program will print the entire ASCII character set.

Bryan Christiansen is a 15-year-old freshman in high school. You may contact him at 314 N. 25th Ave., Fargo, ND 58102.

Machine Language Listing

```
               *       Constants
          LNLNTH  EQU    $1F            Screen line length
          COLOR   EQU    $80            Background color
          SCRSTR  EQU    $400           Scrn mem start address
          SCREND  EQU    SCRSTR+$200    Scrn mem end+$200
                  ORG    $3C35
          STRING  EQU    *-$2           Addr for param passing
               *       Initialize
3C35 86   80      START1  LDA  #COLOR         Get backgroung color
3C37 C6   1F              LDB  #LNLNTH        Get screen line
3C39 8E   0400    LOOP1   LDX  #SCRSTR        Get screen add
               *       Clear left hand edge
3C3C A7   80              STA  ,X+            Clear one loc. on left ed
3C3E 3A                   ABX                 Line increment X
3C3F 8C   0600            CMPX #SCREND        Done?
3C42 2D   F5              BLT  LOOP1          No, continue
               *       Move screen back 1 address
3C44 8E   0400            LDX  #SCRSTR        Get screen add
3C47 A6   01      LOOP2   LDA  1,X            Load char and
3C49 A7   80              STA  ,X+            Move it back one
3C4B 8C   0600            CMPX #SCREND        Done?
3C4E 2D   F7              BLT  LOOP2          No, continue
               *       Print right hand edge
3C50 8E   041F            LDX  #SCRSTR+$1F    Get upper right corner
3C53 FE   3C33            LDU  STRING         Get addr of string to be
                                              printed
3C56 A6   C0      LOOP3   LDA  ,U+            Load char and
3C58 A7   80              STA  ,X+            Print it
3C5A 3A                   ABX                 Increment X to next line
3C5B 8C   0600            CMPX #SCREND        Done?
3C5E 2D   F6              BLT  LOOP3          No, continue
               *       End
3C60 39                   RTS
               *       Clear entire screen (CLS)
3C61 86   80      START2  LDA  #COLOR         Get background color
3C63 8E   0400            LDX  #SCRSTR        Get screen add
3C66 A7   80      LOOP4   STA  ,X+            Clear one location
3C68 8C   0600            CMPX #SCREND        Done?
3C6B 2D   F9              BLT  LOOP4          No, continue
               *       END
3C6D 39                   RTS
```

BASIC Listing

```
1 REM———BANNER BY——————
2 REM———BRYAN CHRISTIANEN———
3 REM———314 N. 25 AVE.—————
4 REM———FARGO, ND  581Ø2———
5 REM———COPYWRITE 11/11/82———
1Ø CLSØ:PRINT@2+32*7,"banner by bryan christiansen";
  :POKE1Ø24+8+32*7,32:POKE1Ø24
  +11+32*7,32:POKE1Ø24+17+32*7,32
2Ø CLEAR256:STUFF$="BANNER BY BRYAN CHRISTIANSEN!!
  PRESS THE SPACE BAR TO ENTER PROGRAM!! "
3Ø EN=15411
34 REM———MACHINE LANGUAGE———
35 REM———SCROLLER——————
4Ø FORA=ØTO44:READB:POKE EN+2+A,B:NEXTA
5Ø DATA134,128,198,31,142,4,Ø,167,128,58,
    14Ø,6,32,37,248
6Ø DATA142,4,Ø,166,136,1,167,128,14Ø,6,
    Ø,45,246
7Ø DATA142,4,31,254,6Ø,5Ø,166,192,167,
    128,58,14Ø,6,Ø,45,246,57
74 REM———CLEAR SCREEN WITH———
75 REM———CHR$——————
8Ø FORA=ØTO12:READB:POKE EN+47+A,B:NEXT
9Ø DATA134,128,142,4,Ø,167,128,14Ø,6,Ø,37,249,57
1ØØ POKE EN+34,INT(EN/256):POKE EN+35,EN-
  INT(EN/256)*256:DEFUSRØ=EN+2:DEFUSR1=EN+47
1Ø5 REM———LETER PARTS——————
11Ø DIMA$(57),B$(59)
12Ø A$(1)="       "
13Ø A$(2)="@@@@@ @"
14Ø A$(3)="@@     "
15Ø A$(4)=" @ @   "
16Ø A$(5)="@@@@@@@"
17Ø A$(6)=" @@  @ "
18Ø A$(7)="@  @  @"
19Ø A$(8)=" @  @@ "
2ØØ A$(9)=" @@   @"
21Ø A$(1Ø)="@  @ @ "
22Ø A$(11)="@  @@  "
23Ø A$(12)=" @@@@@ "
24Ø A$(13)="  @@  @"
25Ø A$(14)=" @ @  @"
26Ø A$(15)="@   @@ "
27Ø A$(16)="     @ "
28Ø A$(17)=" @  @ @"
29Ø A$(18)="@ @@  @"
3ØØ A$(19)="@  @@ @"
31Ø A$(2Ø)="@ @  @ "
32Ø A$(21)="@      "
33Ø A$(22)=" @     "
34Ø A$(23)="  @@@  "
35Ø A$(24)=" @   @ "
36Ø A$(25)="@     @"
37Ø A$(26)="   @   "
38Ø A$(27)=" @ @ @ "
39Ø A$(28)="    @@ "
4ØØ A$(29)="    @@@"
41Ø A$(3Ø)="      @"
42Ø A$(31)="     @ "
43Ø A$(32)=" @     "
44Ø A$(33)=" @    @"
45Ø A$(34)="@     @@"
46Ø A$(35)="@    @ @"
47Ø A$(36)=" @@ @@ "
48Ø A$(37)="   @@  "
49Ø A$(38)=" @  @  "
5ØØ A$(39)="@   @  "
51Ø A$(4Ø)="@@@@  @"
52Ø A$(41)="@ @    "
53Ø A$(42)="@@   @@"
54Ø A$(43)="@@  @@ "
55Ø A$(44)="@@  @@@"
56Ø A$(45)="@ @    "
57Ø A$(46)=" @@    "
58Ø A$(47)="@ @ @ @"
59Ø A$(48)=" @@@ @"
6ØØ A$(49)="   @@@@"
61Ø A$(5Ø)="   @@@ "
62Ø A$(51)="@@@@@@ "
63Ø A$(52)="  @@  "
64Ø A$(53)="     @@"
65Ø A$(54)="@    @ "
66Ø A$(55)=" @@@@ @"
```

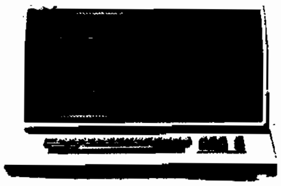
BASIC Listing *(continued)*

```
67Ø A$(56)="@@   @"
68Ø A$(57)="@ @  @"
685 REM———LETTER COMBINATIONS-
69Ø B$(1)="1 1 1 1 1 1 1 1 "
7ØØ B$(2)="1 2 "
71Ø B$(3)="1 3 1 3 "
72Ø B$(4)="1 4 4 5 4 5 4 4 "
73Ø B$(5)="1 6 7 7 5 7 7 8 "
74Ø B$(6)="1 9 1Ø1112131415"
75Ø B$(7)="1 1617187 192Ø17"
76Ø B$(8)="1 2122"
77Ø B$(9)="1 232425"
78Ø B$(1Ø)="1 252423"
79Ø B$(11)="1 2627235 232726"
8ØØ B$(12)="1 2626265 262626"
81Ø B$(13)="1 2829"
82Ø B$(14)="1 26262626262626"
83Ø B$(15)="1 2828"
84Ø B$(16)="1 3Ø163126322221"
85Ø B$(17)="1 12252525252512"
86Ø B$(18)="1 335 3Ø"
87Ø B$(19)="1 3334357 7 7 9 "
88Ø B$(2Ø)="1 24257 7 7 7 36"
89Ø B$(21)="1 31374 38395 31"
9ØØ B$(22)="1 4Ø7 7 7 7 7 15"
91Ø B$(23)="1 127 7 7 7 7 8 "
92Ø B$(24)="1 21213411413 "
93Ø B$(25)="1 367 7 7 7 7 36"
94Ø B$(26)="1 6 7 7 7 7 7 12"
95Ø B$(27)="1 4242"
96Ø B$(28)="1 4344"
97Ø B$(29)="1 264 2425"
98Ø B$(3Ø)="1 4 4 4 4 4 4 4 "
99Ø B$(31)="1 25244 26"
1ØØØ B$(32)="1 2212119454546"
1Ø1Ø B$(33)="1 12257 47473548"
1Ø2Ø B$(34)="1 494 3839384 49"
1Ø3Ø B$(35)="1 5 7 7 7 7 7 36"
1Ø4Ø B$(36)="1 12252525252524"
1Ø5Ø B$(37)="1 5 252525252512"
1Ø6Ø B$(38)="1 5 7 7 7 7 2525"
1Ø7Ø B$(39)="1 5 454545452121"
1Ø8Ø B$(4Ø)="1 12252525257 5Ø"
1Ø9Ø B$(41)="1 5 26262626265 "
11ØØ B$(42)="1 255 25"
111Ø B$(43)="1 163Ø3Ø3Ø3Ø3Ø51"
112Ø B$(44)="1 5 3131264 2425"
113Ø B$(45)="1 5 3Ø3Ø3Ø3Ø3Ø3Ø"
114Ø B$(46)="1 5 3 5231523 5"
115Ø B$(47)="1 5 3 322631535 "
116Ø B$(48)="1 5 25252525255 "
117Ø B$(49)="1 5 454545454546"
118Ø B$(5Ø)="1 12252525355455"
119Ø B$(51)="1 5 4545451110Ø9 "
12ØØ B$(52)="1 467 7 7 7 7 28"
121Ø B$(53)="1 2121215 212121"
122Ø B$(54)="1 513Ø3Ø3Ø3Ø3Ø51"
123Ø B$(55)="1 3 52283Ø28523 "
124Ø B$(56)="1 5 53373237535 "
125Ø B$(57)="1 25244 264 2425"
126Ø B$(58)="1 3 32262926323 "
127Ø B$(59)="1 2534357 575625"
128Ø KOLOR=2Ø1:BACK=169:SPEED=2Ø:
   WIDTH=1:HEIGHT=1:E=4:F=5
129Ø GOSUB13ØØ:GOTO143Ø
1295 REM———LOAD LETTER SET———
13ØØ CLSØ:PRINT@4,"wait for a count of 57";
13Ø1 POKE1Ø24+8,32:POKE1Ø24+12,32:POKE1Ø24+14,32
13Ø2 POKE1Ø24+2Ø,32:POKE1Ø24+23,32:POKE1Ø24+24,32
13Ø3 POKE1Ø24+25,55:PRINT@32:C=EN+6Ø
131Ø FOR A=1 TO 57
132Ø PRINT A;
133Ø FORD=1TOE:POKEC,BACK:C=C+1:NEXTD
134Ø FOR B=1 TO 7
135Ø C$=MID$(A$(A),B,1)
136Ø IF C$="@" THEN CHR=KOLOR ELSE CHR=BACK
137Ø FORG=1TOHEIGHT
138Ø POKEC,CHR:C=C+1
139Ø NEXTG
14ØØ NEXT B
141Ø FORD=1TOF:POKEC,BACK:C=C+1:NEXTD
142Ø NEXT A:RETURN
1425 REM———MAIN LOOP——————
143Ø POKE EN+3,PEEK(EN+6Ø):POKE EN+48,PEEK(EN+6Ø)
   :X=USR1(Ø)
```

BASIC Listing *(continued)*

```
144Ø IF STUFF$=""THENSTUFF$="!"+CHR$(34)+"#$%&'()*+,-.
   /Ø123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ"
145Ø FOR A=1 TO LEN(STUFF$)
146Ø WORK$=MID$(STUFF$,A,1)
147Ø B=ASC(WORK$)-31
148Ø IF B<1ORB>59THENB=1
149Ø FOR C=1 TO LEN(B$(B)) STEP 2
15ØØ D=VAL(MID$(B$(B),C,2))
151Ø D=EN+44+16*D
152Ø POKE EN,INT(D/256):POKE EN+1,D-INT(D/256)*256
153Ø FOR E=1 TO WIDTH:X=USRØ(Ø):NEXT E
154Ø FOR F=1 TO SPEED:IFINKEY$=" "THEN
   PLAY"V31O1T255AA":GOTO158ØELSE NEXTF
155Ø NEXT C
156Ø NEXT A
157Ø GOTO 145Ø
1575 REM———MENUE——————
158Ø CLS1:PRINT@Ø,"          MENUE":PRINT:PRINT
159Ø PRINT"    #1 -CHANGE COLOR OR HEIGHT";
16ØØ PRINT"    #2 -CHANGE WIDTH"
161Ø PRINT"    #3 -CHANGE SPEED"
162Ø PRINT"    #4 -CHANGE MESSAGE"
163Ø PRINT"    #5 -RUN MESSAGE
164Ø PRINT:PRINT:PRINT"    INPUT A NUMBER
        AND PRESS <enter>"
165Ø PRINT:INPUT"WHICH NUMBER???";D
166Ø IF D<1 OR D>5 THEN158Ø
167Ø ON D GOTO168Ø,191Ø,195Ø,199Ø,143Ø
1675 REM———COLOR AND HEIGHT———
168Ø CLS:PRINT"     COLOR AND HEIGHT"
169Ø PRINT:PRINT"    IF YOU KNOW THE NUMBERS YOU
   WANT PRESS <N><ENTER> ELSE PRESS
    <Y><ENTER> FOR HELP."
17ØØ INPUT"DO YOU NEED HELP?";D$
171Ø IF D$<>"Y"THEN182Ø
172Ø I=Ø:CLS:FORH=ØTO255
173Ø POKE1514,H
174Ø PRINT@491,"—————"H;
175Ø PRINT@511," "
176Ø I=I+1:IFI>6THEN18ØØ
177Ø NEXTH
178Ø PRINT@451,"PRESS ANYTHING TO CONTINUE";
179Ø IF INKEY$=""THEN179ØELSE168Ø
18ØØ PRINT@451,"PRESS ANYTHING TO CONTINUE";
181Ø I=Ø:IF INKEY$=""THEN181Ø:ELSE177Ø
182Ø CLS:PRINT"     COLOR AND HEIGHT"
183Ø PRINT:PRINT"    INPUT A NUMBER FOR THE"
1831 PRINT"    LETTERS AND INPUT A NUMBER FOR"
1832 PRINT"THE NUMBERS. THE NUMBERS MUST"
1833 PRINT"BE BETWEEN Ø AND 255. INPUT THE"
1834 PRINT"NUMBERS LIKE THIS"
1835 PRINT" <LETTERS,BACKGROUND><ENTER>."
184Ø PRINT:INPUT"NUMBERS?";KOLOR,BACK
185Ø IF KOLOR<Ø OR BACK<Ø OR KOLOR>255 OR
   BACK>255 THEN PRINT@352,"    NUMBERS
   MUST BE BETWEEN Ø  AND 255":
   FORA=ØTO1ØØØ:NEXT:GOTO182Ø
186Ø CLS:PRINT"     COLOR AND HEIGHT":PRINT
   :PRINT"    DO YOU WANT SINGLE OR
   DOUBLE  HEIGHT? INPUT A <1> OR A <2>."
187Ø PRINT:INPUT"NUMBER?";HEIGHT
188Ø IF HEIGHT<1 OR HEIGHT>2 THEN PRINT@32Ø,
   "  NUMBER MUST BE BE 1 OR 2":FOR A=
   ØTO1ØØØ:NEXT:GOTO186Ø
189Ø IF HEIGHT=1THENE=4:F=5:ELSEE=1:F=1
19ØØ GOSUB13ØØ:GOTO158Ø
19Ø5 REM———WIDTH——————
191Ø CLS:PRINT"          WIDTH"
192Ø PRINT:PRINT"    INPUT A NUMBER FOR THE
   WIDTH.  SUCH AS <NUMBER><ENTER>.";
193Ø PRINT:INPUT"NUMBER?";WIDTH
194Ø GOTO158Ø
1945 REM———SPEED——————
195Ø CLS:PRINT"          SPEED"
196Ø PRINT:PRINT"    INPUT NEW SPEED.  THE
   HIGHER THE NUMBER THE SLOWER
   THE SPEED."
197Ø PRINT:INPUT"NUMBER?";SPEED
198Ø GOTO158Ø
1985 REM———CHANGE MESSAGE———
199Ø CLS:PRINT"     CHANGE MESSAGE"
2ØØØ PRINT:PRINT"    INPUT YOUR NEW MES
   SAGE.   SUCH AS <MESSAGE><ENTER>."
2Ø1Ø PRINT:LINE INPUT"YOUR MESSAGE————";STUFF$
2Ø2Ø GOTO158Ø
```

EAGLE



64K Ram
780 KB Disk Storage
Word Processing, Ultracalc CP/M*
C-Basic Software
Smith Corona TP 1
Letter Quality Printer

$2995.00

EAGLE 1600 .... CALL

TeleVideo

TERMINALS

910	$579.00
912C	$699.00
920C	$749.00
925C	$749.00
950	$950.00
WYSE WY100	$749.00

COMPUTERS

800A	$1299.00
802	$2649.00
802H	$4695.00
806	$4999.00
816	$8999.00
303	CALL
1602, 1603	CALL

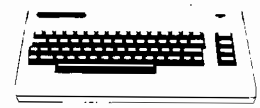
commodore

8032	$1039.00
4032	$749.00
8096 Upgrade Kit	$369.00
Super Pet	$1499.00
2031	$469.00
8250 Dbl.Sided Disk Drive	$1699.00
D9060 5 Meg. Hard Disk	$2399.00
D9060 7.5 Meg. Hard Disk	$2699.00
8050	$1299.00
4040	$969.00
8300 (Letter Quality)	$1549.00
8023	$599.00
4022	$399.00
New Z-Ram, Adds CP/M & 64K	$549.00
The Manager	$209.00
Magis	CALL
Word Pro 5 Plus	$319.00
Word Pro 4 Plus	$299.00
Word Pro 3 Plus	$199.00
The Administrator	$379.00
Info Pro Plus	$219.00
Power	$79.00

commodore

VIC 20

$149



VIC 20 Dust Cover	$9.99
VIC 1530 Datassette	$69.00
VIC 1541 (64K Disk Drive)	$339.00
VIC 1525 Graphic Printer	$339.00
VIC 1210 3K Mem. Exp.	$32.00
VIC 1110 8K Mem. Exp.	$53.00
VIC 1111 16K Mem. Exp.	$94.00
VIC 1011 RS232C Term. Interface	$43.00
VIC 1112 IEEE-488 Interface	$86.00
VIC 1211 Super Expander	$53.00
VIC Mother Board	$99.00

NEC COMPUTERS

8001A	$719.00
8031	$719.00
8012	$549.00

PRINTERS

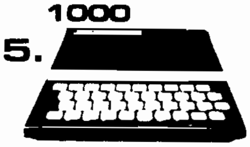
8023	$469.00
7710/7730	$2399.00
3510/3530	$1599.00

MONITORS

JB-1260	$119.00
JB-1201	$149.00
JC-1212	$299.00
JC-1203	$629.00

TIMEX SINCLAIR 1000

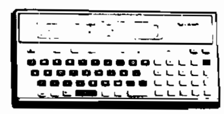
$85.



16K Memory Module	$44.95
Vu-Calc	$17.95
Check Book Manager	$13.95
The Organizer	$14.95
The Budgeter	$13.95
Stock Option	$14.95
Loan & Mortgage Amortizer	$12.95
Mindware Printer	$109.00

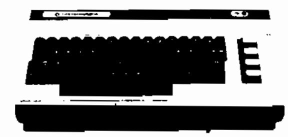
SHARP PC-1500 POCKET COMPUTER

$169



CE 150 Printer, Plotter and Cassette Interface Unit	$172.00
CE 152 Cassette Recorder	$62.00
CE 155 8K Ram Expansion Module	$94.00
CE 125 Printer/Micro Cassette	$129.00

commodore



VIC 64

$429.

PROFESSIONAL SOFTWARE

Word Processing for VIC 64 .... $79.95

MONITORS

AMDEK

100 B & W	$74.95
300G	$169.00
300A	$179.00
Color I	$339.00
Color II	$699.00
Color II A	$799.00
Color III	$399.00
Color IV	CALL

BMC

12" Green	$79.99
13" Color 1401 (Mid Res.)	$369.00
9191U 13"	$329.00

TAXAN

RGB 1	$329.00

ZENITH

ZVM 121	$99.00

SHARP

Sharp 13" Color TV	$275.00

PANASONIC

TR-120MIP (High Res. Green)	$159.00
CT-160 Dual Mode Color	$299.00

PRINTERS

SMITH CORONA

TP 1	$599.00

C. ITOH (TEC)

Starwriter (F10-40CPS)	$1399.00
Printmaster (F10-55CPS)	$1749.00
Prowriter 80 Col. (Parallel)	$499.00
Prowriter 80 Col. (Serial)	$629.00
Prowriter 2 (132 Col.)	$799.00

OKIDATA

82A	$429.00
83A	$659.00
84 (Parallel)	$1049.00
84 (Serial)	$1149.00

IDS

MicroPrism	$649.00
132 (Fully Configured)	$1599.00
80 (Fully Configured)	$1399.00

Call for other configurations.

STAR

Gemini 10	$379.00
Gemini 15	$489.00

DAISYWRITER

Letter Quality	1049.00

DIABLO

620	$999.00
630	$1769.00

MODEMS

HAYES

Smart	$229.00
Smart 1200 (1200 Baud)	$549.00
Chronograph	$199.00
Micromodem 100	$309.00
Micromodem II	$279.00
Micromodem II (with Terms)	$299.00

NOVATION

Cat	$144.00
D-Cat	$159.00
212 Auto Cat	$589.00
Apple Cat II	$279.00
212 Apple Cat II	$609.00
J-Cat	$119.00
Cat 103	$199.00
Cat 103/212	$439.00

ANCHOR

Mark I (RS-232)	$79.00
Mark II (Atari)	$79.00
Mark III (TI-99)	$109.00
Mark IV (CBM/PET)	$125.00
Mark V (OSBORNE)	$95.00
Mark VI (IBM-PC)	$179.00
Mark VII (Auto Answer Call)	$119.00
TRS-80 Color Computer	$99.00
9 Volt Power Supply	$9.00

IBM

NEC 3550 PRINTER...$2099

PERCOM DRIVES

5¼" 160K Disk Drive	$249.00
5¼" 320K Disk Drive	$299.00

AMDEK

310A Amber Monitor	$179.00
310G	$179.00
Amdisk (3¼" Drive)	$729.00
DXY Plotter	$759.00
Color II	$699.00

SOFTWARE

I.U.S. Easywriter II	$249.00
I.U.S. Easyspeller	$129.00
Peach Package (GL/AP/AR)	$419.00

PROFESSIONAL SOFTWARE

IBM/PC Word Processing	$319.00

MICRO PRO

Word Star/Mail Merge	$399.00

computer mail order east

800-233-8950

IN PA. CALL (717)327-9575, 477 E.THIRD ST., WILLIAMSPORT, PA. 17701

In stock items shipped same day you call. No risk, no deposit on C.O.D. orders. Pre-paid orders receive free shipping within the Continental United States with no waiting period for certified checks or money orders. Add 3% (minimum $3.00) shipping and handling on all C.O.D. and Credit Card orders. NV. and PA. residents add sales tax. All items subject to availability and price change. **NOTE:** We stock manufacturer's and third party software for most all computers on the market. Call today for our new catalogue.

Circle No. 4

FRANKLIN

ACE 1000
ACE 10 with Controller Card
ACE Writer Word Processor

CALL...
FOR SYSTEM PRICE!
Ace 1000 .... CALL

SYSCOM II

48K Color Computer
100% Apple Compatible
Apple Soft Basic

$769.00

VISICORP

for Apple, IBM & Franklin

Visidex	$189.00
Visifile	$189.00
Visiplot	$159.00
Visiterm	$89.00
Visitrend/Plot	$229.00
VisiSchedule	$229.00
Desktop Plan	$189.00
Visicalc(AppleII+,Atari,CBM,IBM)	$179.00

Visicorp prices for IBM may vary slightly.

CONTINENTAL

Home Accnt. (Apple/Franklin)	$59.00
Home Accountant (IBM)	$119.00
1st Class Mail (Apple/Franklin)	$59.00

SIRIUS

Free Fall	$24.00
Beer Run	$24.00
Snake Byte	$24.00
Space Eggs	$24.00
Sneakers	$24.00
Bandits	$28.00

BRODERBOUND

Apple Panic	$23.00
David's Magic	$27.00
Star Blazer	$25.00
Arcade Machine	$34.00
Choplifter	$27.00
Serpentine	$27.00

INFOCOM

Deadline	$35.00
Star Cross	$29.00
Zork I	$29.00
Zork II or III	$29.00

MPC

Bubdisk (128K Ram)	$719.00

AXLON

Apple/Franklin 128K Ram	$399.00
Apple/Franklin Ram Disk	$999.00

VU-MAX

80 Column Card	$159.00

PERCOM

DISK DRIVES FOR ATARI

AT 88-S1	$399.00
AT 88-A1	$289.00
RFD 40-S1	$539.00
RFD 40-A1	$329.00
RFD 40-S2	$869.00
RFD 44-S1	$659.00
RFD 44-S2	$999.00

RANA DISK DRIVES

Call for price and availability on the new Rana Disk Drives for The Apple and Franklin Computer Systems.

μ-SCI

MICRO-SCI

DISK DRIVES FOR APPLE & FRANKLIN

A2	$299.00
A40	$349.00
A70	$459.00
C2 Controller	$79.00
C47 Controller	$89.00

FLOPPY DISKS

MAXELL

MD I (Box of 10)	$32.00
MD II (Box of 10)	$44.00
FD I (8")	$40.00
FD II (8" DD)	$50.00

VERBATUM

5¼" SS DD	$26.00
5¼" DS DD	$36.00

ELEPHANT

5¼" SS SD	$19.99

HEWLETT PACKARD

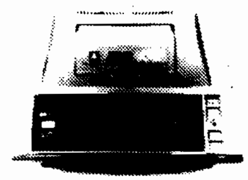
hp 41CV

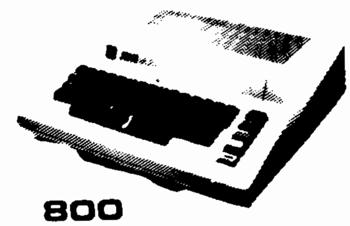
$209

HP 41C	$149.00
HP 10C	$59.00
HP 11C	$72.00
HP 12C	$99.00
HP 15C	$99.00
HP 16C	$99.00

HPIL PERIPHERALS In Stock
Call for
CALCULATOR SPECIALS

ATARI HOMECOMPUTERS





400

16K......... $199
32K........ $274*
48K........ $359*
*Non-Atari Ram

800

48K $499

Call for Price and Availability of the NEW
64K ATARI 1200

410 Recorder	$74.00
810 Disk Drive	$429.00
822 Printer	$269.00
825 Printer	$589.00
830 Modem	$159.00
820 Printer	$259.00
850 Interface	$169.00
CX40 Joy Sticks (pair)	$18.00
CX853 Atari 16K Ram	$77.95

Axlon 32K Ram	$89.00
Axlon 48K Ram	$139.00
Axlon 128K Ram	$399.00
Intec 48K Board	$159.00
Intec 32K Board	$74.00
One Year Extended Warranty	$70.00
CX481 Entertainer Package	$69.00
CX482 Educator Package	$130.00
CX483 Programmer Package	$54.00
CX484 Communicator Package	$344.00

SOFTWARE FOR ATARI

ATARI

Pac-Man	$33.00
Centipede	$33.00
Caverns of Mars	$32.00
Asteroids	$29.00
Missile Command	$29.00
Star Raiders	$35.00
Galaxian	$33.00
Defender	$33.00

ON-LINE

Jawbreaker	$27.00
Softporn	$27.00
Wizard and the Princess	$29.00
The Next Step	$34.00
Mission Asteroid	$22.00
Mouskattack	$31.00
Frogger	$31.00
Cross Fire (ROM)	$36.00

SYNAPSE

File Manager 800	$69.00
Chicken	$26.00
Dodge Racer	$26.00
Synassembler	$30.00
Page 6	$19.00
Shamus	$26.00
Protector	$26.00
Nautilus	$26.00
Slime	$26.00
Disk Manager	$24.00

DATASOFT

Pacific Coast Highway	$25.00
Canyon Climber	$25.00
Tumble Bugs	$25.00
Shooting Arcade	$25.00
Clowns and Balloons	$25.00
Graphic Master	$30.00
Graphic Generator	$13.00
Micro Painter	$25.00
Text Wizard	$79.00
Spell Wizard	$64.00
Bishop's Square	$25.00
Sands of Egypt	$25.00

APX

Text Formatter	$18.50
Family Budgeter	$18.50
Eastern Front	$24.00
Family Cash	$18.50
Jukebox	$13.50
Downhill	$18.50
Outlaw	$18.50
Holy Grail	$24.00
Player Piano	$18.50
Keyboard Organ	$18.50
Number Blast	$13.50
Frogmaster	$18.50
747 Land Simulator	$18.50
Bumper Pool	$13.50

CBS

K-razy Shoot Out	$32.00
K-razy Kritters	$32.00
K-razy Antics	$32.00
K-star Patrol	$32.00

EPYX

Crush, Crumble & Chomp	$24.00
Crypt of the Undead	$24.00
Curse of Ra	$16.00
Datestones & Ryn	$16.00
Invasion Orion	$19.00
King Arthur's Heir	$24.00
Morloc's Tower	$16.00
Rescue at Rigel	$24.00
Ricochet	$16.00
Star Warrior	$29.00
Temple of Asphai	$29.00
Upper Reaches of Apshai	$16.00

WICO Joy Sticks

for Atari, Commodore, Apple & Franklin
CALL

computer mail order west

800-648-3311

IN NV. CALL (702)588-5654, P.O. BOX 6689, STATELINE, NV. 89449

INTERNATIONAL ORDERS: All shipments outside continental United States must be pre-paid by certified check only! Include 3%(minimum $3.00) shipping and handling.
EDUCATIONAL DISCOUNTS: Additional discounts are available from both Computer Mail Order locations to qualified Educational Institutions.
APO & FPO: Add minimum $5.00 shipping on all orders.

CP/M is a registered trademark of Digital Research, Inc.

A Beginner's Computer Glossary

A Beginner's High-tech Glossary
To help you befriend your computer and become literate in its vocabulary, here is a list of commonly used words. Learn the meanings of these words and you will feel more comfortable advancing your computer knowledge.

ASCII — American Standard Code for Information Interchange. A standard 8-bit information code used with most computers and data terminals. Many systems use only seven of the eight bits, providing a total of 128 possible characters including upper- and lower-case alphabet, punctuation, numbers, spacing, and machine or control commands.

Assembly language — A machine-oriented language for programming.

BASIC — Beginner's All-purpose Symbolic Instruction Code. One of the easiest computer programming languages to learn and master.

Bit — 1. A binary digit — either a 0 or 1. 2. A unit of information capacity of a storage device. 3. A single pulse in a group of pulses.

Bug — Any mechanical, electrical, or electronic defect that interferes with the operation of the computer, or a defect in the coding of a program.

Byte — A generic term to indicate a measurable portion of consecutive binary digits. A set of eight bits. A byte is universally used to represent a character. Microcomputer instructions require one, two, or three bytes. A word may be one byte long, or less, or more. One byte has two nibbles.

Chip — An integrated circuit(s) on a wafer slice, usually made of silicon.

CPU — Central Processing Unit. The central processor of the computer system, which contains the main storage, arithmetic unit, and special register groups.

CRT — Abbreviation for Cathode-Ray Tube. The television tube used to display pictures or characters. Also the computer terminal made from a CRT.

Cursor — A position indicator displaying on the video screen the position of a character to be corrected, or a position in which data is to be entered. Different terminals have different restrictions for cursor movements.

Disk — A circular metal plate with magnetic material on both sides, continuously rotated for reading or writing. Disks come in 8"- and 5¼" sizes and are inserted into the computer to hold information.

DOS — Disk Operating System. Operating system integrating disk-file facilities such as symbolic files, automatic space allocation, and sometimes dynamic memory allocation.

EPROM — An electrically programmable ROM suited for high performance systems that need fast turn-around for system program development and for small volumes of identical programs in production systems.

Input/Output — Commonly called I/O. 1. A general term for equipment used to communicate with a computer. 2. Data involved in such communication. 3. The media carrying the data for input/output.

Interface — A common boundary between automatic data-processing systems or parts of a single system, or between two systems or devices.

K (Kilo) — A symbol equivalent to the numeral 1024; e.g., 8K equals 8192.

Loop — 1. The repeated execution of a series of instructions for a fixed number of times. 2. A coding technique for repeating instructions that are usually modified.

Machine language — 1. A set of symbols, character, or signs, and the rules for combining them that conveys instructions or information to a computer. 2. Information or data expressed in code that can be used directly without further processing.

Matrix — A rectangular array of numbers subject to mathematical operation. A table is a matrix.

The glossary will be continued next month.

MICRO

SJB DISTRIBUTORS. THE MOST COMPETITIVE PRICES ON COMMODORE.

commodore

MONITORS - GREAT RESOLUTION (64 OR VIC)

Amdeck Color I	$319
Amdeck II or III	call
Comrex 6500 - 13" Color	299
NEC JC 1212	310
Transtar 20 (High Resolution Green Phosphor)	129
Video/Audio Cable	15

PRINTERS--LETTER QUALITY

CBM 8300, 40 cps	$1450
Diablo 620, 25 cps	995
ComRiter 17 cps	899
Transtar 130, 16 cps (auto load, wp features!)	769
NEC 7700 series	2350
NEC 3500 series	1600

PRINTERS--DOT MATRIX

CBM 4022, 80 cps/graphics	395
CBM 8023, 150 cps/graphics	589
Epson FX Printer	529
Okidata 82A, 120 cps (serial and parallel)	429
NEC 8023A (parallel)	469
IDS Microprism	539
Star Gemini	429
Star Gemini (15 columns)	529
Epson MX 80 Ribbons	6

COMMODORE BUSINESS SERIES

SuperPet (5 languages, 2 processors)	1409
CBM 8032 Computer, 80 Column	1029
CBM Memory Expansion, 64K	359
PET 4032, 40 Column	879
CBM 8050, 1 mg. Dual Drive	1259
CBM 8250, 2 mg. Dual Drive	1500
CBM D9060, 5 mg. Hard Disk	2240
CBM D9090, 7.5 mg. Hard Disk	2600
CBM 4040, 340K Dual Drive	919
CBM 2031, 170K Single Drive	489
DC Hayes Smart Modem	220

BUSINESS SOFTWARE

WordPro 4+ or 5+	309
Administrator	489
VisiCalc (expanded)	199
The Manager (database)	199
BPI A/R, G/L, Job Cost, Inventory, Payroll	325 /ea.

SOFTWARE FOR CBM 64

Word Processing (WordPro 3+)	$69
Word-Pac (tape)	60
The Assistant Series	
Writer's Assistant (easy and flexible)	99
File Assistant (database with merge)	99
Spreadsheet Assistant	99
Personal Finance Assistant (great reports)	45
Coco (computer tutoring game)	44
Coco II (build your own games easily)	45
Home Accounting Package	39
General Ledger A/R, A/P (with check writing)	175ea.
CBM EasyFinance	50
Data Manager	70
Stock (investment analysis)	80
Pet Emulator (emulates 4.0 basic)	30
Sprite-Magic (use joystick to design sprites)	19
Assembler Package (cassette or disk, compiled, includes editor, loader, disassembler)	39
Motormania (arcade tape)	27
Renaissance	27
Colorsketch (draw on screen w/joystick, paint, save to tape or disk)	16
Spacebelt	20
Matchmaker	16
Retroball	34

INTERFACES AND ACCESSORIES

80 Column Expander	159
VIC 1600 Modem	95
VIC 1650 (auto answer, auto dial)	150
VIC 1525 Graphic Printer	329
VIC 1530 Datasette Recorder	65
VIC 1541 Disk Drive	329
VIC Switch (connect 8 64's or Vics to printer,dd)	149
IEEE Interface	85
PET-IEEE cable	33
IEEE-IEEE cable (2m)	39
Parallel Interface (Epson, Okidata, IDS, NEC)	80
RS-232 Printer Interface (Okidata, Diablo, etc.)	60
Apple Loader	90
Programmers Reference Guide	18
Verbatim Diskettes (10 per box)	26
Victree	75

VIC PRODUCTS AND ACCESSORIES

8K RAM Memory Expansion Cartridge	44
16K RAM	85
24K RAM	129
VIC IEEE Interface	75
VIC 3 Slot Expander	39
VIC 6 Slot Expander	70
RS232 Printer Interface	65
Cassette Interface	27
Intro to Basic I or II	22
Home Finance Package (6 tapes)	47
Gorf	30
Omega Race	30
Video Vermin	29
Heswriter (wp cartridge)	29
Turtle Graphics (cartridges)	33
Arcade Joystick - Heavy duty w/2 firing buttons! Great for the VIC or 64	25

TO ORDER CALL TOLL FREE

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

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

Business Hours
M-F 8 to 6
Sat. 10-2

MasterCard, Visa, Money Order, Bank Check
COD (add $5) accepted.
Add 3% surcharge for credit cards.
In stock items shipped within 48 hours, F.O.B. Dallas, TX
All products shipped with manufacturer's warranty.

Prices are subject to change without notice.

Circle No. 42

68000 BCD and Privileged Instructions

by Joe Hootman

The information presented this month was compiled with the assistance of Motorola, Inc.

BCD Instructions

This month I have included a table of BCD instructions (see table 1). These instructions are similar to BCD operations in most microprocessors except that there are only three instructions implemented in the 68000. The basic instruction operates on packed BCD data, stored in memory, in the byte mode. The traditional method used to implement complex operations on BCD data is to convert the BCD to binary, perform the operations on the binary data, and then reconvert to BCD.

Privileged Instructions

The privileged instructions in table 2 are used by the processor when executing programs in the supervisor state. The supervisor state of the 68000 is considered to be a protected state; i.e., the user operating in the user mode does not have easy access to the supervisor state. When in the supervisor state the 68000 can make use of the full instruction set. However, when the processor is in the user state, privileged instructions cannot be executed. If the user attempts to execute any one of the privileged instructions, the processor will go into exception processing. When entering exception processing, the S bit in the status register is set to 1 and the trace bit is set to 0.

Exception processing can be entered two ways — by internal sources or by external sources. Internal sources consist of the following: instructions such as TRAP, TRAPV, CHK, division by 0; privilege instructions used while in the user mode; trace-mode operation; odd addresses associated with long word or word data; and illegal or unimplemented instructions. External sources consist of hardware interrupts, bus errors, or reset request.

Exception processing is used when there may be a problem with either the software or hardware. The processor is designed so that when it is in the supervisor state it can inform the user of possible software or hardware problems, and take the proper steps to correct these potential problems.

Most of the privileged instructions are used to manipulate the user stack pointer and the system status register. There are, however, some special instructions that can be executed in the supervisor state.

Reset External Devices (RESET) is a command that causes the reset pin of 68000 to go low. Any device connected to the reset pin is also reset. This command allows the 68000 to reinitialize the peripheral components using software in the supervisor mode.

Return from Exception (RTE) is the statement that is used to resume processing after an exception has occurred. Generally this statement is used at the end of some service routine to return control to the user.

The STOP instruction is used to stop the 68000 from fetching and executing instructions. Execution does not continue until an external interrupt, trace, or reset exception occurs. The STOP instruction can be used to stop the processor and wait for an interrupt. The immediate data following the instruction is used to set the status register.

***Table 1:* BCD Instructions**

Mnemonic	Data Size/CCR	Name	Comments
ABCD	8 CCR X N Z V C • U • U •	Add Decimal with Extend	The source and destination and the X bit are added together using BCD arithmetic and the result stored in the destination. The X bit is normally set as the result of a previous BCD calculation. Opword Format 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 | 1 | 1 | 0 | 0 | Register Rx | 1 | 0 | 0 | 0 | 0 | R/M | Register Ry | R/M = 0 specifies a data register. R/M = 1 specifies an address register for the predecrement addressing mode (memory to memory). Rx field specifies destination register. Ry field specifies the source register.

(continued)

Table 1 *(continued)*

Mnemonic	Data Size/CCR	Name	Comments
SBCD	8 CCR X N Z V C • U • U •	Subtract Decimal with Extend	The source is subtracted, using BCD arithmetic, from the destination (with the extend bit) and the result is stored in the destination. Opword Format 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 1 0 0 0 Register Rx 1 0 0 0 0 R/M Register Ry R/M = 0 specifies a data register. R/M = 1 Specifies an address register for the predecrement addressing mode (memory to memory). Rx — specifies the destination register. Ry — specifies the source register.
NBCD	8 CCR X N Z V C • U • U •	Negate Decimal with Extend	The addressed data along with the extension bit is subtracted from zero, using BCD arithmetic, and the result is stored in the destination. Opword Format 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 0 1 0 0 1 0 0 0 0 0 Effective Address Mode | Register The effective address field specifies the destination. The following effective addressing modes cannot be used: 2, 10, 11, 12, 13, 14.*

U — the bit is not defined.

***Table 2:* Privilege Instructions**

Mnemonic	Data Size/CCR	Name	Comments
MOVE USP	32 CCR X N Z V C - - - - -	MOVE User SP	The contents of the user stack pointer is transferred to or from the specified address register. Opword Format 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 0 1 0 0 1 1 1 0 0 1 1 0 dr Register dr field specifies the direction of transfer: 0 — specifies the address register to USP. 1 — specifies the USP to the address register. The register field specifies the address register that is used in the transfer.
MOVE to SR	16 CCR X N Z V C • • • • •	Move to the Status Register	The contents of the source operand is moved to the status register. Opword Format 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 0 1 0 0 0 1 1 0 1 1 Effective Address Mode | Register The effective address specifies the destination register. The following effective addresses cannot be used: 2, 10, 11, 12, 13, 14.*
EORI to SR	16 CCR X N Z V C • • • • •	Exclusive OR Immediate to Status Register	The immediate data is exclusive ORed with the contents of the Status Register (SR) and the results stored in the Status Register. Opword Format 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 0 0 0 0 1 0 1 0 0 1 1 1 1 1 0 0 Word Data

(continued)

Table 2 *(continued)*

Mnemonic	Data Size/CCR	Name	Comments
ANDI to SR	16 CCR X N Z V C • • • • •	AND Immediate to Status Register	AND the immediate data with the contents of the Status Register (SR) and place the results in the SR. Opword Format 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 0 0 0 0 0 0 1 0 0 1 1 1 1 1 0 0 Word Data
ORI to SR	16 CCR X N Z V C • • • • •	Inclusive OR Immediate to Status Register	The contents of the Status Register (SR) and the immediate data are inclusively ORed and the results stored in the SR. Opword Format 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 0 0 Word Data
ANDI to SR	16 CCR X N Z V C • • • • •	AND Immediate to the Status Register	The Status Register and the immediate data are logically ANDed and the results left in the Status Register. Opword Format 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 0 0 0 0 0 0 1 0 0 1 1 1 1 1 0 0 Word Data
RESET	—— CCR X N Z V C - - - - -	Reset External Devices	This instruction causes the reset pin of the 68000 to reset the external devices that are connected to the reset pin. Opword Format 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 0 1 0 0 1 1 1 0 0 1 1 1 0 0 0 0
STOP	—— CCR X N Z V C Depends on Immediate Data	Load Status Register and Stop	The immediate data is placed in the SR and the PC is advanced to the next instruction but processing stops at this point. Execution of instructions resumes when a trace, interrupt, or reset condition occurs. Opword Format 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 0 1 0 0 1 1 1 0 0 1 1 1 0 0 1 0 Immediate Data
RTE	—— CCR X N Z V C pulled from stack	Return from Exception	The SR and PC are pulled from the system stack and processing continues at the PC pulled from the stack in either user or supervisor state, depending on the S bit in the Status Register (SR) pulled from the stack. Opword Format 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 0 1 0 0 1 1 1 0 0 1 1 1 0 0 1 1

MICRO™

Learn to program the ATARI™ in 6502 Machine Language & BASIC.
Three new ATARI books for the serious programmer and beginner, are now distributed by IJG, for use with the ATARI 400 and 800 microcomputer systems.
ATARI BASIC,
Using.
This is an action book.
Games For T
Provides ideas on how to c
your own computer games.
How to Program Your ATARI in 6502 Machine Language. To teach the
novice computer user ma
language, the use of an as
to call subroutines
preter. 106 pages.
products are n
your local dealer, or
$4.00 for shipping and
per item. Foreign
If it's from IJG IT'S JUST GREAT!

A Versatile Hi-Res Graphics Routine for the APPLE

by Adam P. King

You can design and use multicolored high-resolution shapes on the Apple. Software is provided for shape collisions, animation, and exclusive-OR drawing.

Graphics Sub
requires:
Apple II with 48K and Applesoft.

One of the deciding factors for my purchasing an Apple II rather than another micro was the Apple's graphics capabilities. The prospect of using the Apple to produce arcade-style games was particularly inviting. Applesoft BASIC comes equipped with its own graphics routines. However, when using these routines to make my own games many inadequacies became apparent. The programs presented in this article use a completely independent graphics routine to remedy these shortcomings.

When designing the routine, I kept the following criteria in mind as essential for arcade-style graphics:

Speed — The routine has to be fast to achieve smooth graphics movement.

Multiple shapes — Many shapes must be able to co-exist. Using different numbers to represent different shapes is ideal.

Collisions — For almost any graphics game it is necessary to know when one shape comes in contact with another shape or object on the screen.

Multicolored shapes — The appeal of multicolored shapes over solid colored shapes is substantial. The flavor of most arcade games would be lost without a diversity of colors.

Constant background — For many arcade-style games, and especially those of the non-shoot-'em-up genre, it is important the drawing of shapes does not destroy previously existing background design.

Applesoft BASIC's graphics routine proved limited on three counts: speed, collisions, and multicolored shapes.

Speed — The Applesoft routines often produce a flicker effect when a shape is moved. This is especially apparent when solid shapes (i.e., not stick figures) are used. The routine presented here is considerably faster and is designed to handle solid shapes. Speed does not come cheaply, however. This routine is relatively inefficient in terms of memory storage so I suggest you use a 48K Apple. Unlike the Applesoft routines, this routine excludes a Rotation and Scale function, thereby gaining additional speed.

Collisions — Applesoft routines do not detect shape collisions. They are extremely difficult to implement within a program using only Applesoft commands.

Multicolored shapes — In the current form of Applesoft graphics, defining a single shape containing more than one color is impossible. Multicolored shapes are achieved only by superimposing one colored shape on another — a slow and tedious procedure.

The routine presented here remedies the above problems and meets the rest of the criteria. In addition, this routine provides an easy-to-implement form of animation.

Entering the Graphics Routine

Entering the graphics routine is a three-part procedure.

1. Enter the BASIC program in listing 1 (Make Tables) and run it. This sets up two tables, which are used by the graphics routine.
2. Enter machine language (call −151) and input the program in listing 2 (Graphics Sub). This is the actual routine.
3. Re-enter BASIC (3D0G) and type "BSAVE GRAPHICS SUB,A$800, L$300".

Enter and save the programs in listings 3 and 4. The first, Shape Definer, allows you to construct shapes. The second, Shape Table Definer, takes these individual shapes and composes a shape table for use with the graphics routine.

Defining a Shape

Before using Shape Definer it is a good idea to draw the shape on graph paper. Any graph paper will do; however, you should note that hi-res "squares" on the Apple are slightly taller than they are wide by an approximate ratio of 1.13:1. Graph paper with this distortion produces more accurate results. Numbered graph paper is ideal because it allows you to identify the odd and even columns (I will explain later). When designing your shapes on paper, fill in squares that will appear as colored dots and leave blank squares to appear black when the shape is drawn. The following six rules provide a guide for defining a shape.

1. Each shape has a Plot Code number. This number is either 1 or 0, depending on whether your shape's leftmost column is drawn on even (0) or odd columns (1).
2. Each horizontal row of your shape has a Color Code that is either 0 or 1. If the Color Code is 0 the row may contain the colors violet, green, black, or white. If the Color Code is 1 the row may contain the colors blue, orange, black, or white.
3. The following table decides the color of an individual dot.

Color Code	Even Columns	Odd Columns
0	violet	green
1	blue	orange

4. Any two horizontal dots placed side-by-side on sequential columns appear white, overiding rule 3.
5. Black appears wherever there is no dot, except in the specific case of rule 6.
6. Two horizontal dots separated by a black space appear to form a solid line the color of the two dots. This rule does not apply if either or both of the dots are white.

An example shape, shown in figure 1, helps clarify these rules and serves as a trial run of the Shape Definer program. This "man" is drawn with his first column placed on an even column in accordance with rule 1. He has a green face, orange neck, violet arms, white body, orange legs, and white feet. To understand how the colors were derived, compare each dot by Color Code and column parity with the six rules.

After your shape has been designed on graph paper it must be translated into data that the graphics routine can use. Shape Definer does most of the work for you, allowing you to input the shape in a form that resembles figure 1.

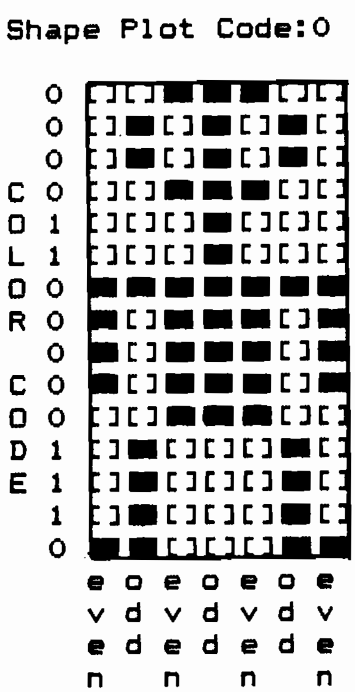


Figure 1: **Shape "man" on graph paper.**

***Listing 1:* Make Tables**

```
5 X =  - 1
10  FOR L = 2304 TO 2559:X = X + 1
20  IF X = 7 THEN Y = Y + 1:X = 0
30  POKE L,Y
40  NEXT L:X =  - 1
50  FOR L = 2560 TO 2815:X = X + 1
60 Y = X: IF  INT (X / 2) = X / 2
     THEN Y = Y + 1
70  IF X = 13 THEN X =  - 1
80  POKE L,Y: NEXT L
90  END
```

When you run Shape Definer, you are asked specific questions about the shape. First you are asked for the shape's width (the maximum width of the shape). In the case of the man, the width is 7. Next you are asked for the length of the shape (the number of horizontal rows that contain the shape). The man's length is 15. Next you must enter the shape's Plot Code. The man's Plot Code is 0. After answering these questions, the information for each row is requested beginning with the top row and working down. For each row you enter the Color Code and then the row itself. The row is entered as a string of 0's and 1's. A 0 represents an empty square on the graph paper and 1 represents a solid square. Figure 2 shows the beginning of a run of Shape Definer using shape "man." After entering the whole shape you are asked if the shape is to be animated. All yes or no questions should be answered with a "Y" or "N". Enter "N" for the man. (How to animate shapes will be discussed later.)

When you finish the shape you are entering, it is relisted and you are asked if the shape is correct. An "N" answer allows you to make changes. In this case you are asked which line is to be changed and you are allowed to change it. When you indicate that the shape is correct, the program takes over. It prints different "PHASES" followed by "WAIT FOR CONVERSION". These listings deal with animation. After seven phases you are instructed how to save the shape. Be careful here; a mistake could cost you the time of re-entering the shape. To make another shape simply run the program again and save the shape under a different name.

Shape Definer can be used to implement a simple, fast, and effective form of animation. When your shape is processed there are actually seven separate versions of the shape saved. These versions, or phases as I call them, correspond to seven relative positions on the

***Listing 2:* Graphics Sub**

```
Ø800-   AD FD Ø8   LDA   $Ø8FD
Ø8Ø3-   8D F9 Ø8   STA   $Ø8F9
Ø8Ø6-   2Ø 88 Ø8   JSR   $Ø888
Ø8Ø9-   AE FE Ø8   LDX   $Ø8FE
Ø8ØC-   BD ØØ Ø9   LDA   $Ø9ØØ,X
Ø8ØF-   8D 81 Ø8   STA   $Ø881
Ø812-   18         CLC
Ø813-   65 26      ADC   $26
Ø815-   85 26      STA   $26
Ø817-   BD ØØ ØA   LDA   $ØAØØ,X
Ø81A-   8D 34 Ø8   STA   $Ø834
Ø81D-   AD FF Ø8   LDA   $Ø8FF
Ø82Ø-   ØA         ASL
Ø821-   ØA         ASL
Ø822-   ØA         ASL
Ø823-   ØA         ASL
Ø824-   8D 2E Ø8   STA   $Ø82E
Ø827-   8D 36 Ø8   STA   $Ø836
Ø82A-   8D 3D Ø8   STA   $Ø83D
Ø82D-   AD ØØ ØC   LDA   $ØCØØ
Ø83Ø-   8D FC Ø8   STA   $Ø8FC
Ø833-   A2 Ø5      LDX   #$Ø5
Ø835-   BD ØØ ØC   LDA   $ØCØØ,X
Ø838-   8D 4B Ø8   STA   $Ø84B
Ø83B-   E8         INX
Ø83C-   BD ØØ ØC   LDA   $ØCØØ,X
Ø83F-   8D 4C Ø8   STA   $Ø84C
Ø842-   AØ ØØ      LDY   #$ØØ
Ø844-   8C FA Ø8   STY   $Ø8FA
Ø847-   AE FC Ø8   LDX   $Ø8FC
Ø84A-   AD 5B ØE   LDA   $ØE5B
Ø84D-   C9 8Ø      CMP   #$8Ø
Ø84F-   FØ 36      BEQ   $Ø887
Ø851-   8D FB Ø8   STA   $Ø8FB
Ø854-   31 26      AND   ($26),Y
Ø856-   29 7F      AND   #$7F
Ø858-   FØ Ø3      BEQ   $Ø85D
Ø85A-   8D FA Ø8   STA   $Ø8FA
Ø85D-   AD FB Ø8   LDA   $Ø8FB
Ø86Ø-   51 26      EOR   ($26),Y
Ø862-   91 26      STA   ($26),Y
Ø864-   EE 4B Ø8   INC   $Ø84B
Ø867-   DØ Ø3      BNE   $Ø86C
Ø869-   EE 4C Ø8   INC   $Ø84C
Ø86C-   CA         DEX
Ø86D-   FØ Ø5      BEQ   $Ø874
Ø86F-   E6 26      INC   $26
Ø871-   4C 4A Ø8   JMP   $Ø84A
Ø874-   EE F9 Ø8   INC   $Ø8F9
Ø877-   AD F9 Ø8   LDA   $Ø8F9
Ø87A-   2Ø 88 Ø8   JSR   $Ø888
Ø87D-   A5 26      LDA   $26
Ø87F-   18         CLC
Ø88Ø-   69 12      ADC   #$12
Ø882-   85 26      STA   $26
Ø884-   4C 47 Ø8   JMP   $Ø847
Ø887-   6Ø         RTS
Ø888-   48         PHA
Ø889-   29 CØ      AND   #$CØ
Ø88B-   85 26      STA   $26
Ø88D-   4A         LSR
Ø88E-   4A         LSR
Ø88F-   Ø5 26      ORA   $26
Ø891-   85 26      STA   $26
Ø893-   68         PLA
Ø894-   85 27      STA   $27
Ø896-   ØA         ASL
Ø897-   ØA         ASL
Ø898-   ØA         ASL
Ø899-   26 27      ROL   $27
Ø89B-   ØA         ASL
Ø89C-   26 27      ROL   $27
Ø89E-   ØA         ASL
Ø89F-   66 26      ROR   $26
Ø8A1-   A5 27      LDA   $27
Ø8A3-   29 1F      AND   #$1F
Ø8A5-   Ø9 2Ø      ORA   #$2Ø
Ø8A7-   85 27      STA   $27
Ø8A9-   6Ø         RTS
```

Listing 3: Shape Definer

```
10  DIM L$(50),CC$(50)
20  HOME : INVERSE : PRINT "SHAPE
     DEFINER": NORMAL
30  PRINT "INPUT WIDTH OF SHAPE";
     : INPUT W1:W2 =  INT ((W1 -
     2) / 7) + 2
40 AC = W1
50  PRINT "INPUT LENGTH OF SHAPE"
     ;: INPUT LE: PRINT "INPUT SH
     APE PLOT CODE";: INPUT CC
60  POKE 8192,W2: POKE 8193,LE: POKE
     8194,CC:AD = 8194
70  HOME : PRINT "BEGIN SHAPE DEF
     INITION": PRINT
80  FOR LO = 1 TO LE
90  IF LO + 2 > 23 THEN  VTAB (24
     ):TA = 23: GOTO 110
100  VTAB (LO + 2):TA = LO + 2
110  PRINT LO;".";
120  HTAB (4): PRINT "COLOR CODE"
     ;: INPUT CC$(LO)
130  IF CC$(LO) <  > "0" AND CC$(
     LO) <  > "1" THEN 90
140  VTAB (TA): HTAB (18): PRINT
     "ROW";: INPUT L$(LO)
150  IF  LEN (L$(LO)) <  > AC THEN
     140
160  IF AG = 1 THEN 210
170  NEXT LO
180  HOME : PRINT "WILL THE SHAPE
      BE ANIMATED";: INPUT A$
190  IF A$ = "N" THEN SK = 1
200  FOR PH = 1 TO 7
210  HOME : INVERSE : PRINT "PHAS
     E #";PH: NORMAL :AG = 0
220  FOR LO = 1 TO LE: IF SK = 1 AND
     PH > 1 THEN 240
230  IF LO = 20 THEN  VTAB (23): PRINT
     "HIT ANY KEY TO CONTINUE LIS
     TING": GET A$: VTAB (21)
240  PRINT LO;".";: HTAB (4): PRINT
     CC$(LO);"  ";L$(LO)
250  NEXT LO
260  IF PH > 1 AND SK = 1 THEN 320
270  PRINT : PRINT "IS PHASE ";PH
     ;" CORRECT";: INPUT A$
280  IF A$ = "Y" THEN 320
290 AG = 1: PRINT "LINE TO CHANGE
      ( 0 TO EXIT )";: INPUT LO
300  IF LO = 0 THEN 220
310  HOME :TA = 2: PRINT LO;". ";
     CC$(LO);"  ";L$(LO): GOTO 110
320  INVERSE : PRINT "WAIT FOR CO
     NVERSION": NORMAL
330  FOR LO = 1 TO LE
340  FOR L1 = 0 TO W2 - 1
350 V1 = 0
360 AD = AD + 1
370 B$ =  MID$ (L$(LO),L1 * 7 + 1,7)
380  FOR L2 = 1 TO 8
390 C$ =  MID$ (B$,L2,1)
400 VA =  VAL (C$)
410 VA = VA * 2 ↑ (L2 - 1)
420 V1 = V1 + VA
430  NEXT L2
440  IF CC$(LO) = "1" THEN V1 = V
     1 + 128
450  IF V1 = 128 THEN V1 = 0
460  POKE AD,V1
470  NEXT L1:L$(LO) = "00" + L$(LO)
480  IF PH = 4 THEN L$(LO) =  MID$
     (L$(LO),9, LEN (L$(LO)) - 8)
     :L$(LO) = "0" + L$(LO)
490  NEXT LO
500 AC = AC + 2: IF AC = W1 + 8 THEN
     AC = W1 + 1
510 AD = AD + 1: POKE AD,128
520  NEXT PH
530  HOME : PRINT "SHAPE DEFINITI
     ON COMPLETED"
540  PRINT : PRINT "COPY OVER THE
      FOLLOWING LINE": PRINT "CHA
     NGING 'XXXXX' TO THE SHAPE N
     AME"
550  PRINT : PRINT " BSAVE XXXXX,
     A$2000,L";LE * W2 * 7 + 10
560  VTAB (5): END
```

Listing 4: Shape Definer Table

```
10  LOMEM: 16384
20  HOME : INVERSE : PRINT "SHAPE
      TABLE DEFINER": NORMAL
30  PRINT "INPUT NUMBER OF SHAPES
      TO ENTER";: INPUT NO: IF NO
      = 0 THEN 30
35  IF NO > 16 THEN 30
40 AD = 3327:A1 = 3071
50 N1 = N1 + 1: HOME : INVERSE : PRINT
     "SHAPE NUMBER ";N1 - 1: NORMAL
60  PRINT : PRINT "COPY OVER THE
     FOLLOWING TWO LINES": PRINT
     "CHANGING 'XXXXX' TO NAME OF
      SHAPE"
70  PRINT " BLOAD XXXXX"
80  PRINT " GOTO 100"
90  VTAB (4): END
100 W2 =  PEEK (8192):LE =  PEEK
     (8193):CC =  PEEK (8194)
110  FOR LO = 1 TO LE * W2 * 7 + 10
120 AD = AD + 1
130  POKE AD, PEEK (8194 + LO)
140  NEXT LO
150 A1 = A1 + 1: POKE A1,W2
160  FOR LO = 1 TO 7:VA = LO
170  IF CC = 0 THEN 200
180 VA = VA + 4
190  IF VA > 7 THEN VA = VA - 7
200 TE = AD - (LE * W2 * 7 + 9) +
     (W2 * LE * (VA - 1) + VA - 1)
210 HB =  INT (TE / 256)
220 LB = TE - 256 * HB
230 A1 = A1 + 1: POKE A1,LB
240 A1 = A1 + 1: POKE A1,HB
250  NEXT LO:A1 = A1 + 1: POKE A1,LE
260  IF N1 < (NO) THEN 50
270  HOME : PRINT "COPY OVER THE
     FOLLOWING LINE": PRINT "CHAN
     GING 'XXXXX' TO FILE NAME"
280  PRINT : PRINT " BSAVE XXXXX,
     A$C00,L";256 + AD - 3326
290  VTAB (3): END
```

Listing 5: Move It

```
20  POKE 16384,0: POKE 16385,0: POKE
     104,64
30  POKE 106,64: POKE 108,64: POKE
     110,64: POKE 126,64: POKE 17
     6,64
40  CLEAR
50  END
```

Listing 6: Test Demo

```
5  HGR :X = 130:Y = 60:OX = 130:O
    Y = 30: POKE 2303,0
10  HCOLOR= 1: FOR L = 0 TO 279 STEP
    30: HPLOT 0,0 TO L,191: NEXT
    L: HCOLOR= 3: FOR L = 0 TO 2
    79 STEP 30: HPLOT 279,0 TO L
    ,191: NEXT L
20  GOTO 40
30  POKE 2301,OY: POKE 2302,OX: CALL
    2048
40  POKE 2301,Y: POKE 2302,X: CALL
    2048:OX = X:OY = Y
50 X = X +  INT (( RND (2) * 3) -
    1) * 4:Y = Y +  INT (( RND (
    2) * 3) - 1) * 4: IF X < 0 OR
    X > 250 OR Y < 0 OR Y > 160 THEN 5
60  GOTO 30
```

horizontal axis that repeat themselves. When a shape is not animated, six phases are simply identical copies of the first phase. If you choose to animate a shape you can edit and change any or all of the seven phases. When editing phases after the first one, it is important to add all leading zeros that the program appends to strings. (This will be obvious when you come to it.) Making small changes in each phase produces animation in a fashion similar to cartoon animation. Repeating form every seven phases is ideal for cyclical motions such as moving arms and legs, blinking lights on a space ship, or spinning propellers on an airplane. The animation is effective only when the shape is moving horizontally — vertical movement does not change the phases. It is important to note that the length and width of an animated shape must be entered initially as the length and width of the largest phase defined.

Using Shape Table Definer

After you save all the individual shapes on disk they must be grouped together into a shape table that allows you to reference different shapes by number, much like the Applesoft shape-table system. When Shape Table Definer is run, you first enter the number of shapes that comprise the table (16 maximum). Then you are instructed how to load the shapes in, one at a time. The number of the shape being loaded appears at the top of the screen. Remember to note which number goes with which shape. When all the shapes are loaded you are instructed how to save the table. This table is BLOADED when you use the routine.

Using the Graphics Routine

The graphics routine is called easily from Applesoft BASIC, Integer BASIC, or assembly language. The three languages have the following in common: the X and Y coordinates for the shape represent the coordinate of the upper left-hand point of the rectangular limits of your shape. This might be a black dot, as is the case with shape "man." All other points are drawn relative to this one. The upper limit of the X coordinate is 255 and 191 for the Y. Because the Apple actually has 280 horizontal positions, there is a space on the right side of the screen, 24 wide by 192 long, on which you cannot plot. This space is ideal (using Applesoft

```
            SHAPE DEFINER

INPUT WIDTH OF SHAPE?7
INPUT LENGTH OF SHAPE?15
INPUT SHAPE PLOT CODE?0

            BEGIN SHAPE DEFINITION

1. COLOR CODE?0   ROW?0011100
2. COLOR CODE?0   ROW?0101010
3. COLOR CODE?0   ROW?0101010
4. COLOR CODE?0   ROW?0011100
5. COLOR CODE?1   ROW?0001000
6. COLOR CODE?    ETC.
```

***Figure 2:* Sample run of Shape Definer.**

graphics) for the game's name, scoring, extra men, etc.

Applesoft: To draw a shape on the screen, the shape's X coordinate should be POKEd into location 2302, the Y coordinate into location 2301, and the shape number into location 2303. If you POKE a shape number that does not have a shape defined for it, a random mess will be drawn on the screen. Similar problems arise when you POKE a Y coordinate, which causes any part of the shape to be drawn off the screen. Try to avoid these situations. Calling 2048 draws your shape, exclusive-OR style. This type of drawing is much like the Applesoft X-Draw command. Calling 2048 twice in succession with the same X and Y coordinate draws the shape first in full color and then erased to black, leaving any original background intact. An important feature of the graphics routine is its collision indicator. If a shape is drawn and a collision with any other non-black dot is detected, a value greater than 0 is stored in location 2298. If P=PEEK(2298) and P takes on a value greater than 0, then a collision has occurred. This value should be PEEKed only after drawing a shape, not erasing a shape.

The graphics routine presented here is completely compatible with Applesoft and Applesoft graphics. However, there is one problem — the graphics routine resides in memory beginning at hexadecimal location $800. As one acquainted with Murphy's Law might expect, this is exactly where Applesoft stores its variables and source code for BASIC programs. Happily, this collision of data can be circumvented. Run the program in listing 5 (Move It) and pointers will be set to locate Applesoft programs and variables above the hi-res screen area. This insures that the graphics routine, hi-res screen, and Applesoft never meet. In Applesoft, Move It must always be run before loading a program, writing a program, loading a machine-language program, or just using Applesoft. If you have written a program that uses the graphics routine, to run it you must do the following (in order):

1. RUN Move It
2. LOAD BASIC program
3. BLOAD Graphics Sub
4. BLOAD shape table being used
5. RUN BASIC program

This procedure could be automated with the use of an EXEC file. Remember, to retain the color integrity of the shape, shapes with Plot Code 0 can have only even numbers for their X coordinate, and shapes with Plot Code 1 can have only odd numbers for their X coordinate. Thus,, when moving a shape around the screen, it must be moved in increments of two along the horizontal axis. The program in listing 6 (Test) demonstrates the graphics routine with shape 0, assuming that the Plot Code is 0. Test moves the shape randomly over a background without erasing it.

Integer BASIC: The Integer routine follows the same POKEs and rules as Applesoft except the Move It program is not necessary. Instead, simply set LOMEM to 16384 before BLOADing the graphics routine and shape table. In Integer there is no inherent command to set the high-resolution mode. The following Integer line acts like the HGR command when executed from within a program.

```
FOR L=8192 TO 16383 : POKE L,0 :
NEXT L : L=PEEK(-16300)+PEEK
(-16297)+PEEK(-16301)+PEEK
(-16304)
```

Substitutiong −16302 for −16301 sets full-screen graphics.

Assembly Language: From assembler, the X coordinate is stored in location $8FE, the Y coordinate in location $8FD, and the shape number in location $8FF. If a collision is detected, a number greater than 0 is stored in location $8FA. To exclusive-OR the shape with the hi-res screen, jump to the subroutine at location $800. Begin all programs after location $4000 to insure that your assembly-language program does not interfere with the graphics.

Advanced Technique

To use the graphics routine optimally you could define all shapes that should register collisions with each other, using only dots of the same column parity, then define a background using only dots of the opposite column parity. This way shapes can run over the background smoothly, detecting only collisions with other shapes. You should not attempt this method unless you have a thorough understanding of the Apple graphics system.

With the graphics routine and supporting software presented in this article, you should be able to make graphics games on the Apple easily. If you take the time to enter and understand the routine, you will find you have a powerful graphics tool at your command. Even the novice programmer should be capable of creating professional-quality graphics.

Adam King is a sixteen-year-old student who taught himself assembly language. He may be contacted at Hut Hill Road, Bridgewater, CT 06752.

MICRO

APPLE Disk Track Copy for Non-Matching Volume Numbers

by Roland E. Guilbault

Change the volume number of your disks for library reference.

I decided to take advantage of the volume numbers on the disks and am writing a program to catalog a disk into a data base file. Up until now I have been using the default disk numbers. Once two or more disks are entered into the data base, it is impossible to tell which disk contains what programs. The volume number on the disk will be used to provide this information. To change my disks over I have to initialize a new disk to the appropriate volume number, then copy the entire disk. To facilitate this process I am using the Call—A.P.P.L.E. disk 6B utilities, which include a track copier. This speeds up the copying process, but the copied disks do not work!

To copy a disk using a track copier, the volume numbers must match. If the volumes do not match, the DOS will not boot correctly.

Embedded within the DOS is the value of the expected volume number. If the expected value does not match the actual disk volume number, the load is ignored. At the end of the boot the loader transfers to an expected code entry and the system errors off. I discovered two places within DOS that should be modified, plus one location in the VTOC. A disk ZAP routine is needed to do these modifications.

Three different sectors must be modified. The first location is in track 0, sector 1. Sector(0,1) must be loaded and then the relative hex location EB changed to the desired volume number. Note: all values in this article are for DOS 3.3. (*Editor's note:* "00" may also be used as a wildcard volume #, since it matches any "found volume.") Sector(0,1) is equivalent to location "B7EB" when the DOS is loaded into a 48K system. This location is in the RWTS parameter list and is the volume number expected. After you modify location "EB" write the sector back out to the disk.

The next location to be modified is in sector(1,9). If this location is not modified your HELLO program will not load correctly, producing a volume mismatch error. No program will be loaded into memory and the cursor will come back ready to accept input. Load sector(1,9) into memory and then set relative location "66" (hex) to the new volume number. This location, equivalent to location "AA66" in the 48K system, is in the DOS key word data. Note that the name of your HELLO program starts at relative location "75" (hex) in this sector. If you want to change the name of the startup program, make the changes at that location on the disk. Now write sector(1,9) back out to disk.

The third location that should be modified is in sector(17,0) in the VTOC table. Load the sector into memory and change relative location "6" to the new volume number. Now write this sector back out to the disk.

In summary, the locations that must be changed to the new volume number are as follows:

1. sector(0,1) displacement EB (hex)
2. sector(1,9) displacement 66 (hex)
3. sector(17,0) displacement 6

Contact Roland Guilbault at P.O. Box 427, Atkinson, NH 03811.

Unfortunately, the procedure described here does not change the volume number of the disk with respect to the "CATALOG" command. All routines that use RWTS derive the volume number from zero-page storage $2F, where RWTS places it. The volume number comes from the Address Block at the beginning of each sector, which is not easily modified. (You may be able to modify it with a nibble editor.) It is possible, however, to have DOS look at the value stored in the VTOC with only a slight modification to DOS, so that a CATALOG will produce the volume number you desire, instead of the actual initialized volume number.

DOS stores the volume number in additional places during initialization.

Track 1	Sector 1	Byte F6	Track 2	Sector 4	Byte BF
Track 2	Sector 2	Byte C1	Track 2	Sector 4	Byte F9
			(two's complement)		

If you want a disk to produce the desired volume number, change Track 1 Sector C Bytes BC and BD to C1 33. This changes the code from LDA $37F6 to LDA $33C1. This modification also requires changing Track 2 Sector 2 Byte AF to AD, or some other ASCII you like; now the CATALOG looks like DISK VOLUME-000 instead of DISK VOLUME@000. Incidentally, the above location is also where you change the name of the volume.

If you don't have a DISK ZAP type of program, the MICRO utility disk now includes an elementary Sector Change program to modify the bytes on a disk. For $10.00 plus $2.00 shipping and handling, you receive: *Applesoft Variable Dump* by Francois, *Straightforward Garbage Collection* by Bongers, *COMPRESS* by Bauers, and *Sector Change* by Daley. The source is not included. Send orders to: Apple Utility Disk, MICRO, P.O. Box 6502, Chelmsford, Massachusetts 01824.

MICRO

SCIENTIFIC SOFTWARE ASSOCIATES, LTD.

Q-card

Questionnaire Analysis Software

- Microcomputer based
  Avoid the expense of contract services -- do everything in-house on your own Apple II+ microcomputer.
- Easy data entry
  Avoid time consuming keypunching. Uses respondent-marked cards entered with an Optical Mark Reader (keyboard entry also possible).
- Comprehensive data analysis
  Sort on any variable(s), tally all responses, conduct cross tabs, correlations, linear regression, frequency distributions, and more.
- Complete editing capabilities
  Weight items, derive composites, add or delete items, and more.
- Easy-to-use
  Programs are user friendly, menu driven, and interactive. No special computer expertise is required.

Call or send for more information today.

SCIENTIFIC SOFTWARE ASSOCIATES, LTD.

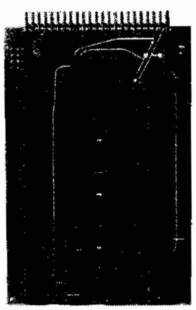
BOX 208 · WAUSAU, WI. 54401
TELEPHONE: (715) 845-2066

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

Circle No. 49

32K CMOS STATIC RAM BOARD for SYM/AIM

Models MB-132/32K, $299
/16K $241,/8K $197

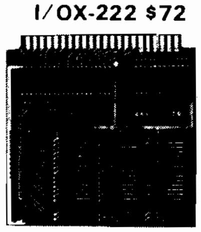


Features:
- 200ns Low Power CMOS, STATIC RAM
- Extends your expansion connector
- Plug compatible with 2716 EPROMS
- First 8K are jumper selectable
- Entire board may be bank-switched
- G-10 Glass epoxy, Full solder mask, Gold fingers
- Full 1-year limited warranty

I/O EXPANSION BOARD for the SYM/AIM

I/OX-122 $60
I/OX-222 $72

and other microcomputers that use 6522 VIAs for I/O and do not provide full address decoding on board. This board has physical space for four additional 6522 VIAs, and provides additional decoding for a total of 16 devices. Connectors for all I/O lines, and further expansion are included. All 6522 functions are available, with no interference with previous functions of the original VIA. Two versions of this board are available. The I/OX-122 mounts above, and directly plugs into, an on-board 6522 socket, and relocates the original VIA to the expansion board. Where there are space limitations, the I/OX-222 uses a dip header and an 8" cable for remote installation.



REAL-TIME CLOCK/CALENDAR $60 Write for Info.
P.O. Box 1019 • Whittier, CA 90609 • (213) 941-1383

ALTERNATIVE ENERGY PRODUCTS

Circle No. 50

What's eating your Apple®?

Find out with Apple-Cillin II™

If you use your Apple for your business or profession, you probably rely on it to save you time and money. You can't afford to guess whether it is working properly or not. Now you don't have to guess. Now you can find out with Apple-Cillin II.

Apple-Cillin II is the comprehensive diagnostic system developed by XPS to check the performance of your Apple II computer system. Apple-Cillin II contains 21 menu driven utilities including tests for RAM memory, ROM memory, Language Cards, Memory Cards, DISK system, Drive Speed, Keyboard, Printer, CPU, Peripherals, Tape Ports, Monitors and more. These tests will thoroughly test the operation of your Apple, and either identify a specific problem area or give your system a clean bill of health. You can even log the test results to your printer for a permanent record.

Apple-Cillin II works with any 48K Apple system equipped with one or more disk drives.

To order Apple-Cillin II - and to receive information about our other products - Call XPS Toll-Free: 1-800-233-7512. In Pennsylania: 1-717-243-5373.

Apple-Cillin II: $49.95. PA residents add 6% State Sales Tax.

XPS

XPS, Inc.
323 York Road
Carlisle, Pennsylvania 17013
800-233-7512
717-243-5373

Apple II is a registered trademark of Apple Computer Inc.

Circle No. 51

It's All Relative, Part 4
Using Commodore's Relative Records

by Jim Strasma

This fourth in our series of articles tells how to actually read and write Commodore relative disk files.

In last month's article we had reached the point of actually using relative files. Read on to learn safe ways to store and read relative file data.

Re-opening a Relative File

First, the relative file is opened for both reading and writing access:

```
1170 DOPEN#1,(F$),D(DD):IF DS THEN
     1690
```

The file name is in variable F$, and the data drive number is in DD. Those with BASIC 2 will need to substitute, as described in part 2 of this series (MICRO 56:53).

Notice that this is (in BASIC 4) exactly the same statement we use to open a sequential file for read access. The file header in the diskette directory tells DOS which type is meant. Also notice the DOS error handler at the end of the line. When an IF-THEN statement is evaluated as true or false, it does so with simple math. Truths are worth −1 and falsehoods are worth 0. If the result of the expression following IF is non-zero, the part of the line following THEN is executed. Otherwise, the program falls through to the next line instead. Thus, IF DS THEN... has the same effect as IF DS 0 THEN... and saves three spaces. Since this statement is needed after nearly every disk command, the savings are considerable. If there were an error, the routine at 1690 closes files, tidies up, and exits to the menu module. Any current changes in the key file are lost, reverting to the contents it had when the current session began. However, nothing is lost from the relative file — all changes, with the possible exception of the one that caused the untimely exit, are preserved.

Adding a Record

Now, as promised, let's tackle our primary goal: writing and reading relative file data. Assuming this is our first use of a newly formatted data file, we need to add some information to the file. In our example mail list, each record is entered in two parts. After making sure there is room for another record, the key field described last issue is entered. Unlike many packages, Bennett's mail list requires each record to have a unique key. When a key is entered, the file is binary-searched for a match. If one is found, the new key is rejected. This protects against re-entering the same name.

If the key is accepted, the user enters the remaining data. All entry is *via* a machine-language editor that filters out troublesome characters. If you enter relative file data without such an editor, do not include quotation marks in the data. They would interfere with the technique used to allow commas, colons, and semicolons (also troublesome) in the data. PET owners may defang quote marks by adding 64 to their ASCII value, making CHR$(34 + 64). But since this doesn't work on the CBM 64, it is often better to just convert quotes into apostrophes.

After a record is entered, the user is asked "Any corrections needed Y/N?". If so, the user is asked to specify a line to change, and that line of data is re-entered in an editing subroutine. Once the data is accepted, the primary and alternate key arrays are updated to include the new record, a currently unused record number is assigned to it, and the record is written to disk.

Here is the subroutine used to write out the data:

```
4960 REM WRITE RECORD FROM D1$
4970 RECORD#1,(RR)
4980 IF DS THEN 1690
4990 D$ = K$ + C$
5000 FOR I = 1 TO NF − 1:D$ = D$ +
     QT$ + D1$(I) + C$:NEXT
5010 D$ = D$ + QT$ + D1$(NF)
5020 PRINT#1,D$;
5030 IF DS THEN 1690
5040 RETURN
```

First note that the record command parks the disk head at the correct spot in the file for this entry, as described in part 2. If there is a disk error here or elsewhere in the subroutine, a panic exit is made back to the menu module, *via* line 1690. As mentioned before, BASIC 2 users will need to call a disk error checking subroutine from lines 4980 and 5030, and use an alternate form of the RECORD command in line 4970.

In line 4990 we begin to build a string of information to write to the disk. The first piece is the key field, in K$, followed by a carriage return character, in variable C$. Every other field in turn is concatenated onto the end of the data string (D$). Each data field is preceeded by a quote character (in QT$), and each field but the last is followed by a carriage return. The quote character allows all other characters to be read back later using an INPUT# statement. Otherwise, we would need the much slower GET# statement.

You'll find a similar advantage when using carriage returns. Since each field is shorter than 80 characters, the INPUT# statement can read them, provided that each ends with a carriage return. This also allows longer fields, without lengthening the overall record. When carriage returns separate fields, there is no need to know the location of each field within the record. Just be sure fields are written and read in the same order. This allows the record to appear larger on the screen than on disk. In Bennett's program, if every field were filled, the record couldn't hold the resulting information. This is rarely a problem, because a full field normally alternates with a barely-used one, balancing out overall. The cost of the technique is the space wasted by the carriage returns.

No carriage return is added to the last field because the disk already knows when the last character is reached. The IEEE-488 EOI (end or identify) line flashes. This changes the ST status variable to 64, and any current INPUT# finishes, just as though a carriage return *had* been added. The added logic in lines 5000 and 5010 that handle this do slow the program a bit. If speed is crucial, add a carriage return at the end, and skip the special handling.

We only PRINT# the data to disk after the entire record has been concatenated into the single variable, D$. Each record must be written with a single PRINT# statement. Otherwise each succeeding PRINT# will go to the next record sequentially — almost never what we have in mind. Even preceeding each successive PRINT# to the same record with a RECORD# statement wouldn't help. In that instance, each new PRINT# to the record would overwrite earlier PRINT#s. This could be avoided only by using the byte extension to the RECORD# statement for each PRINT#, specifying where within the record to begin the current write, described in part 2. If you use the extension, be sure each successive PRINT# begins after all earlier PRINT#s within the record. Otherwise, those following the new PRINT# will be overwritten. (Each PRINT# always writes to the end of the record, regardless of where within the record it begins the write.)

After the disk write subroutine, our new information is on disk in record #RR. However, the mail list could forget that the record is on disk if the program halted before re-writing the key file. It is a good idea to update keys about once an hour when adding to a file. Safeguard changes whenever you've made enough that you don't want to redo them.

Changing a Record

Now let's assume we have several entries in a mail list, and we want to read one. This is done with Bennett's change command (even if we have no changes to make). It begins by asking for the record's key field. Enter as much as you remember; the nearest match will then be found. If it is not an exact match, you will be warned "*** Not Found ***", but even so, it will be retrieved from disk for your viewing. You may then browse alphabetically through the file, pressing the [UP-ARROW] key to see the next record, and the [BACK-ARROW] key to see the prior record. Here are two subroutines that help:

```
6220 REM CALC NEXT RECORD
6230 IF K9<NV THEN K9=K9+1
6240 RR=K%(K9)
6250 RETURN
6260 REM CALC PRIOR RECORD
6270 IF K9>1 THEN K9=K9-1
6280 RR=K%(K9)
6290 RETURN
```

The key position of the current record is in K9. Going up, if it is less than the last active record (NV), then K9 is increased and RR is assigned the alphabetically next record number to read. Going down, if it is greater than 1, the first record, (K9) is decreased, and RR is assigned the prior record.

To be sure the record will fit the space allotted, each new or altered record has its length checked:

```
5410 REM RECORD LENGTH CHECK
5420 ER=0
5430 WK=LEN(K$)+1
5440 FOR M=1 TO NF:WK=WK+
     LEN(D1$(M))+2:NEXT
5450 WK=WK-1
5460 IF WK>RL THEN ER=1
5470 RETURN
```

Variable ER serves as an error flag. It starts equal to zero, but if the record is too long, it is changed to one. After the subroutine, the main program checks the value of ER to see if the record is too long. If so, the user must redo the last field altered, to make it shorter.

If we wrote an overlong record to disk, the disk copy would be truncated at the assigned length, losing remaining data. On the other hand, records may safely be shorter than the allotted space.

One other check is made on newly entered postal code data. If the postal code is not valid in either the U.S. or Canada, the program accepts it, but displays a warning. (Bennett's original version made the user re-enter such data.)

Finally we are ready to read a relative record:

```
4890 REM READ RECORD INTO D1$
4900 RECORD#1,(RR)
4910 IF DS THEN 1690
4920 INPUT#1,K$
4930 IF DS THEN 1690
4940 FOR I=1 TO NF:INPUT#1,D1$(I):
     IF DS=0 THEN NEXT:RETURN
4950 GOTO 1690
```

Through 4930, the program is like the key field reader we used last month. The rest of the work is done in line 4940. Unless an error occurs, it completes the record read and returns to the main program. The reason for writing it this way is to save time. When a FOR-NEXT loop is entirely contained on one BASIC line, the BASIC interpreter doesn't waste time looking for line numbers. In a long program, the savings are considerable. Line 4950 will be executed only if there is a disk error.

Occasionally the program may halt at line 4940 with a ?STRING TOO LONG ERROR. This means over 79 characters were read without a carriage return. A retry with the same record always succeeds. Unfortunately, if it happens, the program will halt. Move the cursor to a blank line and type:

```
GOTO 1190
```

to return to the update menu without losing data.

Other Features

Before leaving the update module, consider its other useful features. First, if some data fields are left entirely blank when entered, the program fills them with default contents, as selected by the start-up module. This speeds data entry when many records share similar information. This typically includes the town and state name, the zip code, and the leading part of the phone number. Here's an example:

```
2270 IF I=4 AND L$=""
     THEN L$=D3$
```

If field number four is empty, this fills it with default #3, as defined in the start-up module.

Second, if the user botches a record and wants to abort the change mode, this is done by pressing [SHIFT + RETURN] when asked to "Select Field for Updating (1 to 8)". The line that does this is:

```
2680 IF FS=0 OR W9$=SR$
     THEN 2780
```

This line skips the update when the last character W9$ equals SR$, preset to [SHIFTED-RETURN]. It also skips when nothing needs updating, as indicated by the flag variable FS still being equal to zero.

Third, the mail list allows the user to make a quick easy copy of either the entire current record, or just the mail label portion of it, by pressing a normally unused key, selected in the set-up module. Here's the line that calls for a mail label:

```
2960 IF G$=E2$ AND PZ$<>"N"
     THEN GOSUB 6490:REM PRINT
     MAIL LABEL
```

E2$ has previously been set to whatever key you want to have trigger the label dump, and the check of PZ$ makes sure you do have a printer before using it.

Fourth, hitting [RETURN] alone usually returns the user to the prior menu, thus giving an easy out if you wander into the wrong section of the program.

In addition to the delete command mentioned last time, there is also a way to change a record's key field. This works by deleting the record, but saving the information, and then immediately re-entering it under the new key name. Like the delete command, it makes sure the user is serious about the change before going ahead with it.

Perhaps the most crucial commands of all in the long run create and retrieve a sequential copy of the entire relative file. This accomplishes at least two goals. It allows the user to restructure the mail list, with differing field lengths, and record lengths, without losing data. Beyond this, if the user ever upgrades to a commercial mail list program, most are able to add data from a sequential file. Therefore you usually won't have to retype information. This is a crucial feature of serious file-handling programs. Using the same technique, this information can also be fed into word processing programs, allowing the user to write customized letters to large lists.

Here is the subroutine that dumps the entire file in sequential order:

```
3470 FOR II=1 TO NV
3480 : RR=K%(II)
3490 : IF G$="R" THEN RR=AL%(II)
3500 : PRINT "DUMPING RECORD"
     II" OF"NV
3510 : GOSUB 4900:REM READ
     RECORD
3520 : PRINT#5,K$
3530 : FOR JJ=1 TO NF
3540 : : PRINT#5,QT$D1$(JJ)
3550 : NEXT
3560 NEXT
3570 IF DS THEN 1690
3580 FOR II=1 TO 9
3590 : PRINT#5,EF$
3600 NEXT
3610 DCLOSE#5
3620 IF DS THEN 1690
```

Note line 3490. Actually, the file can be dumped in two orders — alphabetical by key, or by increasing record number, using the alternate key. Dumping a file in alphabetical order allows you to keep rarely-changing lists in very neat order. This is sometimes useful when using the print module, as we will see next time. However, if you use the record numbers for anything, such as envelope numbers in a church, you won't want them changed, and should always dump in record number order.

Loss of Power

Even if power goes off to the computer, current relative file data is usually preserved, with the key file reverting to its prior contents. On the 8050 and later drives, a surge protector within the disk system preserves the integrity of diskette data during loss of power. On the 4040 and 2040 however, a power loss often causes faulty data to be written momentarily. Depending on where the read/write head is at the time, this could be disastrous. If your area is prone to such outages, make regular backups, suspect any diskette in use during a power outage, and keep the drive doors open when not accessing the disk (i.e., no files open). Better yet, buy a backup power supply for about $450. Be sure it protects the disk drive in addition to the computer.

Multiple Users

In Bennett's mail list, the relative file remains open until the user is done with the module. Usually this is no problem. However, leaving it open full-time does create a problem for those wanting to connect two or more computers to the same disk drive and have everyone use the same file. To do this safely, the file would be opened just before information from it is needed and closed again immediately after the data is read or written. This ensures data in the DOS buffers gets written to disk, so all users are working with the same information. If you will have multiple users, you may also want to add a special one-character field to the record itself. This would be a busy signal to other users. If it has one value, it would mean it is not in use, and anyone may use it. If it has another value, someone is looking at that record. In that case, no one else should be allowed to change it. Otherwise, there could be two versions of one record active at once.

Single Drives

To accommodate the needs of those with only a single small disk drive, "Update" begins by noting whether two drives are to be used:

```
1070 IF PD<>DD then 1110
```

PD is the drive to use for programs, and DD is the drive to use for data. If these are the same, "Mail List" assumes the user has a 2031/1541 disk drive. In that case, the user is prompted to remove the program diskette, insert a data diskette, and then press a key. Those with D90 series CBM hard disk drives may want to change this line to always GOTO line 1110. (Those drives are big enough to hold both programs and data, and there is no diskette to remove.)

BASIC 4 for Everyone

As a final note I offer an alternative for those with VICs, 64s, and older PETs. Richvale Telecommunications (10610 Bayview, Richmond Hill, Ontario, Canada L4C 3N8) and Skyles Electric Works (231 E. South Whisman Road, Mountain View, CA 94041) sell plug-in BASIC 4 equivalents. RTC supplies them on plug-in cartridges, called "V-Link" for the VIC and "C64-Link" for the 64. A Skyles ROM for PET is called "Disk-O-Pro" and a Skyles product for VIC is called "VicTree." I can't imagine using my CBM 64 without my C64-Link; it certainly eases the work in adapting large programs, such as Bennett's mail list, originally written for PET BASIC 4. First reports on the Skyles products are also favorable.

Contributing editor Jim Strasma is assistant professor of computer science at Lincoln College. You may reach him at 1280 Richland Ave., Lincoln, IL 62656.

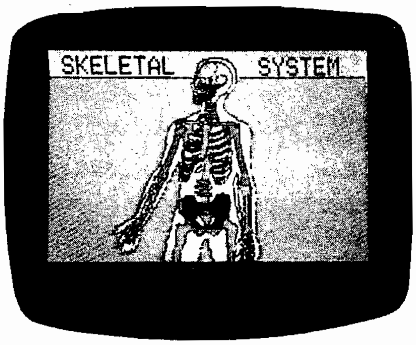
MICRO

Circle No. 56

FOR COMPLETE GRAPHICS:

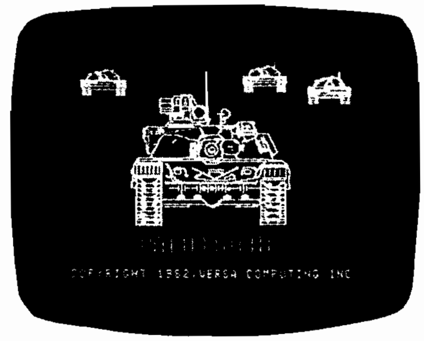
VersaWriter

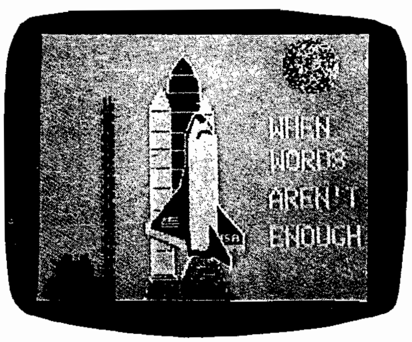
EDUCATION



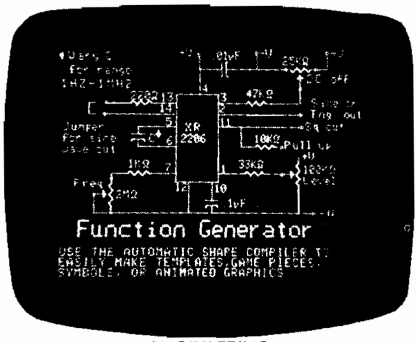
ARTIST

GAME PROGRAMMER

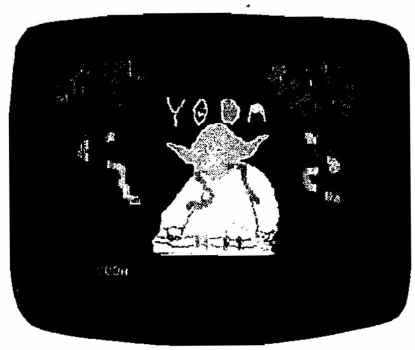




HOBBIEST

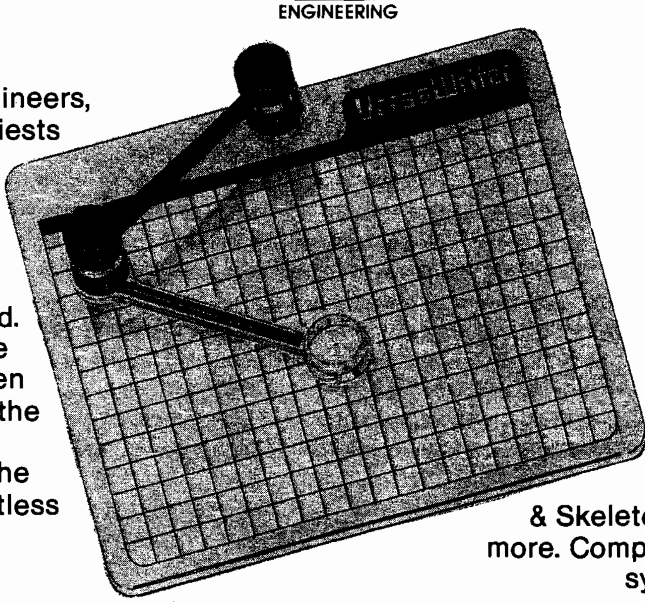


ENGINEERING



CHILDREN

Teachers, artists, engineers, programmers & hobbiests find VersaWriter an easy to use tool for creating micro computer graphics. No programming experience is required. Pictures can be made by simply tracing. Even children can explore the exciting world of computer graphics. The VersaWriter is as limitless as your imagination.



VersaWriter contains complete software for drawing with color, brushes & dots. Add text or fill in over 100 colors. Create your own shapes and place anywhere on the screen. Use Area/Distance, Move Picture, Electronic Drawing & Skeleton programs plus much more. Complete hardware/software system for Apple II/II+/IIe - **$299.00**

VERSA COMPUTING, INC.

Versa Computing Products are available at your local computer products store.

VersaWriter is also available with software designed for Atari & IBM PC.

Distributed by:

Computerland Corp.
Hayward, Calif.

Softsel Computer Products
Inglewood, Calif.

Pete & Pam Computers
Lancashire, England

Micron Distributing
Toronto, Canada

Program Spektrum
Bromma, Sweden

Micro Products Sales Group
Lynn, Mass.

Educational Media
Washington, Penn.

ESD Laboratories
Tokyo, Japan

Blue Ridge Computers
Capetown, South Africa

3541 Old Conejo Road, Suite 104 • Newbury Park, CA 91320 • (805) 498-1956

PET Vet

By Loren Wright

ANDs and ORs, PEEKs and POKEs

VIC and Commodore 64 owners may be a little puzzled about all the mysterious programming required to control video and sound on their machines. Most of this is accomplished with four BASIC functions, and understanding how they work will make things a lot easier.

PEEK and POKE are fairly easy, since their names actually describe what they do. PEEK means you are looking at a particular memory location and returning with the number found. PEEK(7680) returns the number found in memory locaton 7680. POKE works the other way around; you take a number and stick it into a specified memory location. POKE 7680,0 puts an '@' character (the one represented by 0) in the upper left corner of an unexpanded VIC screen. The numbers involved may be BASIC expressions, as long as the addresses are kept between 0 and 65535 and the numbers are kept between 0 and 255. Don't go POKEing numbers into memory at random; many combinations will cause your computer to crash.

All the control registers of the VIC, the VIC II, and the SID are treated by the computer as normal memory locations. You can program a great number of the chips' functions using PEEKs and POKEs with these memory locations. If you look carefully at the VIC Chip portion of this month's VIC Data Sheet (page 103), you will see that controlling the chip is not as simple as POKEing a number into an address. For instance, $900F (36879) actually controls three different things: the screen color, reverse mode, and the border color. How do you change the screen color without affecting reverse mode or the border color? One way is to know what those are supposed to be and POKE the appropriate number. But you do need a way to figure out what the border color and reverse mode statuses are.

Each memory location is actually a byte consisting of eight bits. Each one of these bits can be either on (1) or off (0). Together they make up a binary number, which converts to decimal by adding in increasing powers of two, starting at the right end. If the right-hand bit is on you add in 2 to the 0 power, or 1. This continues 2, 4, 8, 16, 32, 64, until you reach 128 (or 2 to the 7th power) at the left end. If there is a 1 in the binary number, then add in that power of two. The bits are numbered according to the power of two they represent.

The 16 possible screen colors for the VIC all can be represented in four bits. Color 0 is represented by 0000, while color 15 is represented by 1111. To fit into bits 7 to 4 of the VIC register, you must multiply the color by 16, which is the same as sticking four 0's at the right end of the binary number. For color 2 (or red) you would use 16*2, or 32. Now you know what number to stick in. However, if you just POKE 36879,32, you will get a red screen with a black border and all the characters reversed. You can preserve the original border color and reverse mode status with the AND function. AND is a Boolean function usually used in a BASIC IF...THEN statement (i.e., IF X = 5 AND Y>0 THEN GOSUB 500). If both conditions on either side of the AND are satisfied then the whole expression is true; otherwise the expression is false. PET BASIC, unlike most other BASICs, allows the AND function to work on the bit level. Like the BASIC expressions above, a bit is considered true if it is 1 and false if it is 0. The two numbers compared by the AND function are compared bit by bit at each bit position. If both bits at a given position are 1, then the resulting number will have a 1 there; otherwise that bit position will get a 0. Consider the following example:

```
    10101011
AND 00001111
    --------
    00001011
```

The equivalent BASIC statement is '171 AND 15'. In PET BASIC, this equals 11, the result of this bit-level AND operation. In other BASICs, it would equal 1 or -1, indicating that both 171 and 15 were non-zero and therefore 'true.' Notice that by ANDing with 15, you have preserved bits 3-0 exactly the way they were in the first number. If you wanted to preserve bits 7-4, you would use 240, or '11110000'. instead. ANDing with 15 is exactly what you want to do to preserve the border color and reverse-mode status. Combining this with the color (calculated above) involves the OR operation.

The OR operation also is used most often in BASIC IF...THEN statements, such as 'IF X$ < "A" OR X$ > "Z" THEN PRINT "NOT A LETTER": RETURN'. If either (or both) of the expressions separated by the OR is evaluated as 'true,' then the overall expression is also true. Only if both are false, is the result false. PET BASIC's OR also operates on the bit level. If either of the corresponding bits in the two numbers is 1, then the result will have a 1 in that position.

```
   00001011
OR 00100000
   --------
   00101011
```

This example has successfully combined the screen color (red = 2) with the existing border color (cyan = 3) and reverse-mode condition (off = 1). To make sure that this works on a general example, I will run through it step by step.

```
PEEK(36879) = xxxxxxxx
(may be any number 0 to 255)

xxxxxxxx AND 15 = 0000xxxx

yyyy * 16 = yyyy0000
(yyyy-any color number 0 to 15)

0000xxxx OR yyyy0000 = yyyyxxxx

POKE 36879, yyyyxxxx
```

If Y is a color number 0 to 15, then one BASIC expression will change the screen color without disturbing the border color or the reverse mode: POKE 36879, PEEK(36879) AND 15 OR (16*Y).

Frequently you want to change just one bit in a register. For instance, bit 7 of VIC register 36874 acts as an on/off switch for the VIC's voice 1. To turn this bit on without affecting the frequency: POKE 36874, PEEK(36874) OR 128. To turn the bit off: POKE 36874, PEEK(36874) AND 127.

MICRO

ARK COMPUTING INC.

Your Salvation In The Sea Of Inflation.

P.O. Box 2025
Corona, CA 91720

SPECIALS

CP/M

AMDEK	Microsoft Premium System	Novation Smart Cat
12" Amber Monitor	Includes: softcard, ramcard, videx Videoterm, softswitch, user guide.	1200 Baud full duplex.
List: 210.00	List: 695.00	List: 595.00
ARK: 164.95	**ARK: 525.00**	**ARK: 495.00**

ASSEMBLERS

	List	ARK
A.L.D.S.	125.00	89.95
LISA	79.95	59.95
LISA ED PAK	119.95	79.95
MERLIN	64.95	54.95

COMMUNICATIONS

	List	ARK
ASCII Expresso Pro	129.95	99.95
DATA Capture 4.0	64.95	49.95
VISITERM	100.00	79.95
Z TERM PRO	149.95	109.95

DATABASES

	List	ARK
dBASE II	695.00	499.95
DB MASTER	229.00	159.95
DB Utility #1	99.00	69.95
DB Utility #2	99.00	69.95
PFS	125.00	89.95
VISIFILE	250.00	184.95

EDUCATIONAL

	List	ARK
Know Your Apple	34.95	29.95
Mastertype	39.95	29.95
Speed Reading Courseware	99.95	79.95
Typing Tutor II	24.95	19.95

FINANCIAL MODELING

	List	ARK
Calestar	145.00	109.95
Desktop Plan II	250.00	184.95
Visicalc	250.00	184.95

GRAPHICS

	List	ARK
The Artist	79.95	59.95
EZ Draw	49.95	39.95
GAS	75.00	54.95
Graphics Magician	59.95	44.95
Hires Secrets	125.00	89.95
Visiplot	200.00	159.95

HARDWARE

	List	ARK
Applicard 6mhz	595.00	395.00
Graphics Plus	149.95	99.95
Keyboard Plus	99.95	69.95
Kraft Joystick	64.95	49.95
Lower Case Plus	64.95	49.95
Lower Case Plus II	24.95	14.95
Micromodem II w/term. pak	409.00	319.95
Microsoft 16K Card	99.95	79.95
Microtek Parallel Interface	139.50	64.95
Videx Combo	425.00	250.00

MAILING LISTS

	List	ARK
Address Book	49.95	39.95
1st Class Mail	74.95	54.95
Magic Mailer	69.95	
Magic Mailer	69.95	54.95
MailMerge	250.00	189.95

PRINTERS

	List	ARK
C. Itoh Prowriter 8510	595.00	475.00
C. Itoh Prowriter 1550	995.00	750.00
C. Itoh Starwriter F10	1395.	1500.
NEC NEC 8023	695.00	475.00

UTILITIES

	List	ARK
Apple Doc	39.95	29.95
Applewriter Preboot	19.00	15.00
The Dictionary	99.95	79.95
E P F IV	79.95	59.95
Disk Recovery	30.00	21.95
Dos Boss	24.00	17.95
Dosource	39.95	24.95
Locksmith	99.95	69.95
Super Disk Copy III	30.00	21.95
Tip Disk #1	20.00	14.95
Utility City	29.50	21.95
Visicalc Preboot	49.95	39.95
Watson	49.95	39.95

WORD PROCESSORS

	List	ARK
Exec. Sec.	250.00	189.95
Magic Window	99.95	79.95
Magic Window II	149.95	119.95
Pie Writer	149.95	99.95
Screenwriter II	129.95	79.95
Super Text 40/80	175.00	129.95
Word Handler	199.00	149.95
Wordstar	495.00	369.95

IF YOU DON'T SEE WHAT YOU NEED, GIVE US A CALL AND WE'LL TRY TO GET IT FOR YOU!

CALL FOR FREE CATALOG
(714) 735-2250

We accept VISA/MASTERCARD, Personal checks (allow 10 days to clear) or COD (2.00 charge). Please include 3% for shipping (2.00 min.) or 5% for blue label (3.00 min.). Calif. residents add 6% sales tax. All items are new and carry manufacturers warranty. Prices and availability are subject to change without notice.

Product Name: **Foxygraf (tape version)**
Equip. req'd: TRS-80C, 16K
Price: $29.95 cassette
$34.95 disk
Manufacturer: Computerware
4403 Manchester Ave.
Encinitas, Ca 92024

Description: *Foxygraf* is a graphics development program for the TRS-80C. Its major function is a screen drawing program that allows simple selection of page and screen. Graphic screens can be easily interfaced with machine-language routines. Files may be stored on tape.

Pluses: Written in relocatable machine code, *Foxygraf* is compatible with the RS disk system. In addition, memory moves and jump to machine-language subroutines are allowed. A HELP command displays all available options. The program teaches much about the 6847 Video Display Generator (VDG).

Minuses: None noted.

Documentation: A 56-page manual provides complete documentation of the program including major entry and patch points. A discussion of the VDG is included.

Skill level required: Beginners will have no trouble using the screen drawer, but a knowledge of assembly programming is required to get the most from the software.

Reviewer: John Steiner

Product Name: **Turtle Graphics**
Equip. req'd: Any VIC
Price: $40.00
Manufacturer: Human Engineered Software
71 Park Lane
Brisbane, CA 94005

Description: An easy-to-learn graphics language along the lines of LOGO.

Pluses: The easily installed cartridge offers few commands and uses letters instead of whole words. *Turtle Graphics* is a painless way to learn to write short programs. The manual covers the commands well.

Minuses: VIC's high-resolution drawing capability is totally ignored, which limits resolution to 22 by 24 dots.

Skill level required: Even pre-schoolers should be able to learn this language.

Reviewer: Jim Strasma

Product Name: **The Arithmetic Classroom: Decimals**
Equip. req'd: Apple II Plus or Apple II (with Applesoft), DOS 3.3
Price: $49.95
Manufacturer: Sterling Software
Sterling Swift Publishing Co.
7901 South IH-35
Austin, TX 78704
Authors: Patricia M. Mullinix, David N. McClintock, and Fawzy T. Ibrahim of Courses By Computers, Inc.

Description: *Decimals* is one of a series of eight programs covering addition, subtraction, multiplication, division, fractions, and decimals. This objective-based tutorial has plenty of practice exercises but is not just a drill program. At the end of each lesson is a mastery test.

Pluses: The tutorial approach allows each type of problem to be explained first. Good feedback is provided; correct answers prompt encouraging messages, incorrect answers prompt hints. The program also covers converting words ("two tenths") to numeral format.

Minuses: Program covers converting mixed fractions to decimals, but only fractions with powers of ten denominators.

Documentation: Well-written manual has clear instructions, practice (paper) exercises with answer key, and achievement record forms. Disk programs are easy to use.

Skill level required: An understanding of fractions. No programming required.

Reviewer: Mary Gasiorowski

Product Name: **Software Development System SDS80C**
Equip. req'd: TRS-80 Color Computer with printer
Price: $89.95
Manufacturer: The Micro Works
P.O. Box 1110
Del Mar, CA 92014

Description: This product is a complete Editor, Assembler, and Debug Monitor in a program pack for the Color Computer. ABUG, the debug monitor, is a short version of CBUG, their excellent stand-alone monitor. It has memory examine/change, display/change registers, two ways of entering a user program, plus cassette utilities. The powerful line-oriented screen editor has been optimized specifically for editing assembly-language source code. Editor features include twelve cursor move options, string search, line insert/delete, plus block move/copy. The full-feature assembler section has conditional assembly, evaluation of complex expressions, and optional direct-object tape generation. The assembler provides a variety of printer options and allows single-step screen assembly.

Pluses: A maximum of software development is given in a minimum of time. The assembler is the equal of most professional assemblers, allowing the experienced programmer to use all "tricks of the trade" in developing 6809 software.

Minuses: A few minor facets would be disturbing to an experienced programmer; i.e., a program made up of unconnected modules must be handled as separate modules and then installed as separate blocks of object code.

Documentation: The 40-page booklet is clear and thorough. In addition, the booklet gives a concise discussion of 6809 programming techniques, including a good primer on position-independent code.

Skill level required: Novice assembly-language progammer.

Reviewer: Ralph Tenny

Product Name:	**Alphabet Squares**
Equip. req'd:	Apple II with Applesoft, 48K RAM, and DOS 3.3
Price:	$29.95
Manufacturer:	Versa Computing, Inc. 3541 Old Conego Rd., Suite 104 Newbury Park, CA 91320

Description: Young children can practice initial consonant and vowel sounds using keys, paddles, or joystick to move a happy-face cursor to one of the three hi-res pictures of letters. The correct one chosen is then redisplayed full-screen. This continues until all letters have been correctly matched.

Pluses: Kids like it and it reinforces letter sounds. Attractive graphics use color intermixing for varied shades and textures. Not copy protected; modifiable BASIC code.

Minuses: More could be done to maintain enduring interest. Music is marginal. Paddles usable but awkward; keyboard OK; joystick is best. Not protected against RESET or CTRL-C.

Documentation: Durable, plastic-covered storage folder is attractive and adequate.

Skill level required: Ability to recognize letters by sight.

Reviewer: Jon R. Voskuil

MICRO

THE ROUTINE MACHINE™

SEARCH

GRAPHICS

A Versatile Programming Utility for the Apple II.

PRINT USING

SOUND

SORT

Now, from the programming experts at S.D.S., an easy-to-use way of putting the POWER and SPEED of machine language routines in YOUR OWN APPLESOFT PROGRAMS!

ROUTINE MACHINE does all the work for you — no knowledge of machine language programming, whatsoever, is required. Simply choose the routine you need from an ever-growing library, and Routine Machine will effortlessly put them in your program, and all done transparently! No need to be aware of or bother with BLOAD's, HIMEM:, etc.

Best of all, with just this starter package, you'll have the routines to put High Resolution **graphics** and **sound** in your programs immediately! Also included is a versatile **print using** module to banish the "decimal point demons" forever! To round out the package, we've also included powerful **search** and **sort** routines (for single dimension arrays; Search: 1000 elements in 1 second Sort: 1000 elements in 90 seconds), and a number of other often-needed routines as well (30 routines in all).

Additional library disks titled **"Ampersoft Program Library"** are already available.

Some of the other routines in The Routine Machine (plus others not listed) are:

SWAP: Swaps two string or numeric values.

TEXT OUTPUT: Prints with no "word break" on screen.

STRING OUTPUT: Input any string, regardless of commas, etc.

ERR: Stack fix for Applesoft ONERR handling.

GOTO, GOSUB: Allows computed statements. Example: **GOTO X ✱ 5** or **GOSUB X ✱ 5.**

BLOAD: Load any binary file 5 times faster than normal. Hi-Res pictures load in under 2 seconds.

RESET HANDLER: Treats RESET with ONERR; or will RUN or reboot disk.

HI-RES ASCII: Character set for mixing text Hi-Res graphics.

TURTLE GRAPHICS: Versatile Hi-Res graphics routines for easy drawing of Hi-Res figures.

OUR GUARANTEE

IF YOU DON'T SAVE MORE THAN THE PURCHASE PRICE OF 'ROUTINE MACHINE' IN YOUR OWN PROGRAMMING TIME IN THE FIRST 30 DAYS YOU OWN IT, SIMPLY RETURN IT FOR A COMPLETE REFUND, NO QUESTIONS ASKED!

southwestern data systems™

P.O. BOX 582 • SANTEE, CALIFORNIA 92071 • TELEPHONE: 714/562-3670

MICRO™

Software Catalog

Name: **Stock Momentum Studies**
System: Apple II
Memory: 48K
Language: BASIC, assembly language
Hardware: Two disk drives, printer, 2K EPROM

Description: The user can chart any stock, commodity, or index in a variety of modes: differentials (momentum), moving averages, exponential averages, percentage changes, or multiple overlays. It permits all parameters and time frames up to one year of daily data adjustable to user's requests.

Price: $525.00
Includes 2K EPROM, system master, data, two communications disks, and manual.

Author: Kenneth Troy, Harrison Folan

Available:
Troy-Folan Productions Inc.
29 Miller Road
Wayne, NJ 07470

Name: **C64 File and Filewriter/Filereader**
System: Commodore 64
Language: BASIC
Hardware: Datasette recorder, printer optional

Description: *C64 FILE* is a multi-purpose database management system that will allow the user to construct, sort, maintain, and print out a relatively wide range of data types. *Filewriter/Filereader* allows the user to produce data files on cassette tape from the information typed in and [illegible] any data files stored on cassette tape.

Price: $9.95 - C64 File
$6.95 plus $2.00 shipping - Filewriter/Filereader
Includes software cassette and documentation. Also includes data tape for C64 File.

Author: Kinetic Designs
Russell Grokett

Available:
RAK Electronics
P.O. Box 1585
Orange Park, FL 32073

Name: **Data Fax**
System: Apple II, Apple II Plus, Apple III, IBM PC
Memory: 64K
Language: Pascal
Hardware: One disk drive

Description: This free-form relational filing system is modeled after the standard folder-page "manual" system (i.e., manilla folders, etc.). It is simple to use to keep track of unstructured information.

Price: $249.00 - Apple 80-column
$199.00 - Apple 40-column
$299.00 - IMB PC
Includes 250-page manual.

Available:
Link Systems
1640 19th Street
Santa Monica, CA 90404
(213) 453-1851

Name: **The Menu Maker**
System: Apple II Plus
Memory: 48K
Language: Applesoft
Hardware: One disk drive minimum

Description: The *Menu Maker* will help you create and edit disk program menus. It allows up to 30 items per menu and up to 15 sub-menus plus full editing: add, delete, change, and rearrange display order. The *Menu Maker* generates a BASIC program you can modify.

Price: $19.95
Includes disk and user notes.

Author: Larry Houbre, Jr.

Available:
L.R.H. Enterprises
358 Ashley Blvd., lw
New Bedford, MA 02746
(617) 997-7346

Name: **HBJ Computer SAT®**
System: Apple II, Apple II Plus
Memory: 48K
Language: BASIC

Description: Now your Apple can help you cram for the Scholastic Aptitude Test (SAT). *Computer SAT* is a computer package that leads students step-by-step through the test preparation process. The program diagnoses the student's strengths and weaknesses, prepares a study plan, and guides him/her through a comprehensive set of study exercises.

Price: $69.95
Includes 470-page textbook, "How to Prepare for the SAT," 50-page user's manual, two double-sided diskettes.

Available:
Harcourt Brace Jovanovich, Inc.
1250 Sixth Avenue
San Diego, CA 92101
and leading bookstores and computer stores

Name: **Gusher™**
System: Apple II, Apple III, IBM PC
Memory: 64K - Apple II and IBM PC
128K - Apple III
Language: Pascal
Hardware: Two disk drives and printer and interface (132-column)

Description: This specially designed accounting package for oil- and gas-well operators automates joint interest billing and revenue distribution. It calculates revenue distribution from production runs for each revenue owner, joint interest statements for all working-interest owners, and A.F.E. reports. The package also generates well pay-out reports and tracks the balances of revenue and working-interest owners. Invoices from and payments to vendors are also tracked. Additional reports provided include 1099 reports, list of expense categories, list of vendors and vendors' invoices. The package also prints checks.

Price: $995.00
Includes complete documentation.

Available:
High Technology Software Products, Inc.
P.O. Box 14665
2201 N.E. 63rd St.
Oklahoma City, OK 73113
(404) 478-2105

Name: **Super Image Saver/Printer (SISP)**
System: PET/CBM 2001, 3000, 4000, 8000
Memory: 16K RAM minimum
Language: BASIC original, Upgrade, or 4.0
Hardware: Any Commodore graphics printer w/programmable line spacing (optional)

Description: If you like or need to draw pictures, charts, schematics, diagrams, etc., on the PET/CBM screen using PET's graphics, but can't stand typing in the program, then *SISP* is for you! *SISP* lets you draw simple or complex full-screen pictures, save, print, modify, and recall them with ease. The program comes fully documented and steps the user through the complete process.

Price: $9.95 cassette
$14.95 on 4040/2031 disk

Author: Louis F. Roehrs

Available:
TELE-TREX Software Systems
4 Waring Lane
Littleton, CO 80121
(303) 770-8144

Name: **The Dimensional Analysis of the Great Pyramid**
System: Apple II
Memory: 48K
Language: Applesoft
Hardware: One disk drive

Description: Programs include modern and ancient measurements of the three chambers, principle triangles, and coffer. Volume magnitudes are emphasized. The programs are a mixture of theory and history, based on 25 years of research. Programs on Archimedes' Cattle Problem and Rhind Mathematical Papyrus are included.

Price: $25.00
Includes diskette.

Available:
Louis K. Bell
P.O. Box 7
Augusta, GA 30903

(continued)

Software Catalog *(continued)*

Name: **SpeedSTAT (Volume 1: Frequencies and Crosstabs)**
System: Apple II or Apple II Plus
Memory: 48K
Language: Applesoft or 6502 Assembler
Hardware: Two disk drives
Description: An extremely easy-to-learn, easy-to-use statistics package designed for non-programmers that has a capacity of over 10,000 data points and over 30 different statistics measures. *SpeedSTAT* is a state-of-the-art, general purpose business and marketing analysis tool.
Price: $250.00
Includes diskettes, user-written manual, cloth-covered slipcase.
Author: Shaffer & Shaffer
Available:
SoftCorp International
229 Huber Village Blvd.
Westerville, OH 43081
800-543-1350 (toll free)

Name: **Data Manager**
System: VIC-20, COM-64
Memory: 8K expander
Language: BASIC
Hardware: Disk drive and printer
Description: *Data Manager* is a complete data management system with up to 1200 entries on a disk. You may define your own data then add, change, or delete any record from your disk. You may also search, print out a hard copy of any or all fields, and run totals.
Price: $59.95 - VIC-20
$79.95 — COM-64
Includes documentation and binder.
Available:
MicroSpec, Ltd.
2905 Ports O'Call Ct.
Plano, TX 75075
(214) 867-1333

Name: **Job Costing/Tracking**
System: Apple II, Apple II Plus, or Apple III with emulator (Applesoft in ROM)
Memory: 48K
Language: Applesoft
Hardware: Two disk drives, 132-column printer
Description: Contractors can save time estimating, writing proposals, analyzing and tracking job costs. The program is remarkably easy to use and is designed to provide numerous valuable reports and financial information for any size contractor.
Price: $237.00
Includes program diskettes and instruction manual.
Author: Jeff Park
Available:
Software Solutions
9124 Hwy. 17
Scotts Valley, CA 95066
(408) 438-2433
or from any dealer

Name: **Bulk Mailer**
System: Apple II
Memory: 48K
Language: Applesoft
Hardware: Single or dual disk or Corvus hard disk
Description: A professional quality mailing-list program that features duplication elimination, fastest possible sorts, unlimited coding capability, multiple label and default options, and instant access to any record. Diskette version (up to 2400 records) can later adapt to hard disk version (up to 32,000 records).
Price: $125.00 - diskette
$250.00 - Corvus hard disk
Includes complete documentation
Author: Joe Marinello
Available:
Satori Software
5507 Woodlawn N.
Seattle, WA 98103

Name: **MicRo Math Blaster**
System: TRS-80 Model III, VIC-20
Memory: 16K - TRS-80
8K - VIC-20
Language: BASIC
Description: *MicRo Math Blaster* offers a combination of arcade game excitement and basic mathematic drill in addition, subtraction, multiplication, and division with levels of difficulty from grades one through eight.
Price: $15.95
Includes instruction manual.
Available:
M-R Information Systems, Inc.
P.O. Box 73
Wayne, NJ 07470

MICRO™

MICRO™

Hardware Catalog

Name: **Color Chart**
System: Commodore CBM/PET Rockwell AIM
Description: *Color Chart* is a color-video RAM board that operates in eight different modes — alphanumeric 32 × 16 with built-in character generator to high resolution 128 × 192 pixels. *Color Chart* generates an independent RS170 output. Business graphs and game graphics can be displayed in color.
Price: $139.95
Includes color chart board and instructions.
Available:
CGRS Microtech
P.O. Box 102
Langhorne, PA 19047
(215) 757-0284

Name: **Mannesmann Tally Microprinter MT160**
Memory: 2000 characters
Language: USASCII plus six European languages
Description: This 80-column serial matrix printer has 160 characters per second friction feed, optional tractor feed, dot addressable graphics, optional correspondence quality print with right-margin justification, text centering, and proportional spacing. The printer features 8-bit parallel and RS232 interface.
Price: $845.00 and up
Available:
Mannesmann Tally
(for OEM quantities)
Industrial distributors, computer dealers, and computer stores

---

Name: **Rememory Board**
System: AIM 65
Memory: 4K
Description: The *Rememory Board* for the AIM 65 provides 4K of low power CMOS RAM with battery backup. Simply remove the AIM 65 RAM chips and plug the *Rememory Board* into the RAM sockets. Programs and data are protected if the computer is turned off or loses power, and are ready to run when the power is restored. No special programming, rewiring, or soldering is required.
Price: $220.00
Includes 4K *Rememory Board* with battery.
Available:
Devices and Services Co.
10911 Dennis Road
Suite 405
Dallas, TX 75229

Name: **DISCOVER**
Description: *DISCOVER* is a stylish and sturdy diskette box expertly crafted of naturally finished solid wood with brass simulated aluminum that provides convenient and safe storage for as many as forty-five 5¼" diskettes in three equally spaced sections. The low profile top design keeps the lid out of the way when open and conveniently ready to close with one hand.
Price: $59.95 Redwood; $67.95 Oak; $69.95 Walnut plus $3.00 s/h
Available:
Leland Young Company
P.O. Box 4127
Bay Pines, FL 33504

Name: **RB5X**
System: All
Memory: 8K
Hardware: Mobile Intelligent Robot
Language: Tiny BASIC
Description: *RB5X* is an experimental intelligent robot that learns from its own experience, progressing from simple random responses to an ability to generalize about features of its environment. Equipped with its own microprocessor, memory, programs, and tactile sensors, the *RB* is intended as a tool for experimenters. It can be "trained", studied, and adapted in a variety of ways according to the imagination of its owner. With the standard RS-232 interface, the robot's memory can be studied, and its programs altered *via* a personal computer. A Special Option Package with 16K add-on memory, Polaroid Rangefinder sonar sensor, and pulsating light option is also available. Additional options under development include a mechanical arm, a voice synthesizer, and digital radio communication between *RB5X*s.
Price: $1195.00
Special Option Package, $295.00
Available:
RB Robot Corporation
14618 W. 6th Ave.
Suite 201
Golden, CO 80401
(303) 279-5525

Name: **ZVX4 Megabyter**
System: Apple II Plus
Hardware: Apple II Plus with 48K-56K, one 5¼" drive
Language: Apple DOS 3.3, Pascal, CP/M (Microsoft and PCPI)
Description: *ZVX4 Megabyter* is an 8" disk controller for Apple II that provides up to 2.2 megabytes with two double-sided/double-density disk drives. It uses IBM industry-standard 3740 format to create standard 8" diskettes that are usable on other computer systems.
Price: $445.00
Includes controller, manual, startup software.
Available:
Sorrento Valley Associates, Inc.
11722 Sorrento Valley Road
San Diego, CA 92121
(619) 452-0101

Name: **Rover 1 Portable Computer**
System: Dual 8 Bit Microprocessors (6502 Family)
Memory: 128K RAM, 4K PROM, built-in tape drive
Language: 6502 Assembly, BASIC and others shortly
Description: The *Rover 1 Portable Computer*, a part of the Rover Computer System, is suited for business applications, home, office, or schools. Standard full/travel keyboard with ten function keys, 110V or 220V, RS-232, serial and parallel ports, real-time clock, video output, accepts CP/M Cards, uses both floppy and hard disk drives. The Rover System weighs less than 18 lbs. The battery will allow up to six hours of operation with the Multifunction Expansion Unit (option 001).
Price: $1850.00 Computer
$2500.00 System
Includes Rover 1 Portable Computer, Rover Display Unit, Multifunction Expansion Unit (option 001) and attache carrying case.
Available:
Exclusive Dealer Distributer
M.P. Computer Services Corp.
2396 Encinal Station
Sunnyvale, CA 94087
(408) 735-0871
Dealer inquiries invited

Name: **Rabble OZI Expansion Board**
System: Ohio C1P Superboard 11
Memory: 32K RAM plus up to 32K ROM
Description: The *Rabble Expansion Board* system for the C1P contains 32K CMOS RAM and sockets for 32K ROM. A simple modification to the C1P will permit 40K RAM (using C1P 2114's). Features are: Ohio compatible floppy disk controller with motor control and data separator, real-time interrupt facilities, two programmable sound generators and amplifier provide six simutaneous sounds to be output at once, and user in/out expansion is provided for with a peripheral and versatile interface adaptor (PIA and VIA). A prototype area is available for approximately 20 ICs.
Price: $430.00
$90.00 bare board and manual
Includes fully socketed, fully populated (except ROM) board tested and set up for 5¼" drives (8" if requested N/C). Ninety-page manual with circuits.
Available:
Rabble OZI Computers
P.O. Box 781
Shepparton VIC 3630
Australia

MICRO™

Language Packages

Editor's Note: This list provides a sample of the most popular packages available and is not meant to be comprehensive.

FORTH

At	Extended fig-FORTH	Atari	10K plus	$100
At 400/800	FORTH	Quality Software	24K, 810 drive	$79.95 disk/manual
6809	FORTH	Kenyon Microsystems	TSC FLEX 9.0 DOS	$100
Ap II	FORTH	Kuntze		$90
At 400/800	valFORTH Int'l.	valFORTH	24K plus 10-12K working space	$45 plus $40 Gen.U til. & Vid. Ed.
TRS-80C	Color-FORTH	Hoyt Stearns Electronics		$123 (RAM & ROM), $58.95 (RAM)
TRS-80C	Color-FORTH	Hoyt Stearns		$58.95 (RAM), $123 (ROM & RAM)
Ap II/II	FORTH 72, Version 2	MicroMotion	1 drive	$89.95, $139.95 hi-res & floating point
Ap II	MicroSPEED I, II (structured, totally recursive, compiled language)	Applied Analytics	48K RAM, 1 drive	$495/$645
Ap II/II	FORTH 1.7	Information Unlimited Software	APDOS, 48K RAM, 1 drive	$140
Ap II/II	OGI fig-FORTH	On-Going Ideas	48K, Disk II	$40 diskette or $60 with OGI Pascal
Ap II, 6809, 6502	fig-FORTH source listings	3 FORTH Interest Group		$10
Ap II/II	polyFORTH	FORTH, Inc.	48K, Z80 card	Level 3 ($600), Level 4 full system source ($3200)

FORTRAN

Ap II/II Plus	Apple FORTRAN	Apple	48K with 16K RAM card/Ap Pascal Lang. Sys/2 drives	$500
Ap II/II Plus	FORTRAN-80	Microsoft	48K/16K RAM card optional/1 drive - 2 preferred/ W.M. Softcard includes CP/M	$195 program alone plus $400 for softcard
Ap	FORTRAN	Compu/Think	PILOT	$69.95
Ap	Nevada FORTRAN		CP/M	$199.95
OSI	FORTRAN 3.4	OSI	CP/M, 8 disk drives	$400
Ap	FORTRAN-77	SofTech Microsystems, Inc.	48K RAM/64K for Prog. Develop., UCSD p-System	$375 for 8 disk plus $375 for UCSD p-System
AP II/II	Nevada FORTRAN	Ellis Computing	Softcard, CP/M, 1 drive	$29.95

LISP

Ap	LISP	Owl Computers		
Ap	APP-L-ISP	Datasoft, Inc.	DOS 3.2, 1 disk drive	$124.95
Ap II/II	p-LISP	GNOSIS	APDOS, 48K RAM, Applesoft in RAM or Lang. Card	$149.95 disk (tutorial & manual incl.)
6502/Ap	muLISP/ muSTAR-80	Microsoft	APDOS, 32K, 1 drive, CP/M	
At 400/800	INTERLISP	Datasoft, Inc.	48K RAM, 1 drive	$124.95

LOGO

Ap II/II Plus	LOGO	Krell Corp.	disk drive (1)/lang. card	$99 no frills combination/ $89 frills only/$149.95 combination
TRS-80C	TRS-80 Color LOGO	Radio Shack	1 disk drive, 32K	$99 disk
Ap II/II Plus	Terrapin LOGO	Terrapin, Inc.	1 disk drive/lang. card, 16K memory board	$149.95
Ap II/II Plus	Apple LOGO	Apple	1 disk drive/lang. card, 64K RAM	$175
VIC-20	Turtle Graphics	HES		$39.95

Micro-Dymanic

Ap II	Micro-Dynamic	Addison-Wesley Publishing Co.	2 disk drives, lang. or RAM card & Pascal	$245

Modula-2

Ap II	Modula 2 (systems implementation)	Volition Systems	Apple Pascal 1.1	$495
68000	Multi-FORTH	Creative Solutions	64K (Motorola MEX68KDM)	$1500-5000

MUMPS

6809, 68000, Apple	MSM-09	Micronetics Design Corp.	64K min.	$695

This Language Package information sheet is continued from the February issue.

Language Packages

Pascal

At	Atari Pascal	Atari	disk drive	$49.95
6809/Ap	Dynasoft Pascal 1.2	Dynasoft Systems Limited	6809-16K/cassette. Ap-disk/cassette	$45 cass (6809), $55 disk/ $50 cass (Ap), $90 (1.3) GIMIX systems with FLEX
CB/M	Tiny Pascal	Abacus		$40 cass/$35 disk
6809/6502/68000	UCSD Pascal	SofTech	64K for Prog. Development/UCSD p-System	$375
Ap II/II Plus	Pascal-Level I	On-Going Ideas	Disk II/DOS 3.2 or 3.3/48K/Applesoft ROM	$35 diskette (Level I)/$69 (Level II) generates 6502 machine code
Ap	Pascal	Merrimack Systems		
CB/M	Pascal	Commodore Business Machines		$250
6809	P-6800 Pascal	Lucidata	GIMIX under TSC FLEX	$90 (2.1 MiniFLEX), $100 (2.2 FLEX 2), $100 (FLEX 9)
TRS-80C	Color Pascal	Computerware	ASCII Text Editor/32K min.	$49.95 (tape), $59.95 (disk)
6809	UCSD Pascal	3 Tallgrass Technologies Corporation		$495
Ap	Pascal Compiler	Microware	OS9	
Ap	Pascal Compiler	Omega Software	OS9/M DOS/FLEX/XDOS/DOS69	$425
Ap	JRT Pascal	JRT Systems, Inc.	CP/M	$29.95 with manual
68000	Pascal	Hemenway Assoc., Inc.		
Ap	Tiny Pascal Plus	Abacus	32K ROM or 48K RAM	$55 cass/$59 disk with DOS
PET	KMMM Pascal	AB Computers	16K or 32K, disk drive	$85 disk
Ap	Apple Pascal	Apple	recom. 2 drives	$495
OSI C4PF/CP-C4	Pascal 1.3	OSI	C4PF - 5¼ disks, CP-C4 - 8 disks	$460
Ap	Pascal/M 2.2	Sorcim	CP/M, Z-80 card, 64K, 2 drives	
CBM	Tiny Pascal	Abacus		$49.95
Ap II	Pascal I	On-Going Ideas	48K ROM Applesoft, 8K min. runtime	$35
Ap II	Pascal II	On-Going Ideas	48K, Disk II	$60
Ap II	Program Development I (OGI Pascal Level I)	On-Going Ideas	48K ROM Applesoft, Disk II	$60

PILOT

Ap II	PILOT	On-Line	Z-80 card	$150
Ap	APPILOT	Muse		$49.95
PET	PETPILOT	Commodore		$24.95
Ap	Nevada PILOT M			$149.95
COM	Vanilla PILOT	Tamarack	16K RAM	$29.95
Ap	APPILOT II (EDU-DISK)	Muse	48K, DOS 3.2	$99.95
Ap	Apple PILOT	Apple	recom. 2 drives	$150
Ap II/II	Nevada PILOT	Ellis Computing	Softcard, CP/M, 1 drive	$29.95
At	Atari PILOT	Atari		$179.95

Small C

6809	Small C Compiler	Wordsworth	FLEX 9 or DOS 69D, 48K	$52.90 (FLEX 9),$47.50 (if you have RLOAD), $47.50 (DOS69D)
TRS-80C	Small C Compiler	Duggers Growing Systems	Assembler Pkg.	$59.95 plus assembler pkg.

VIGIL

VIC/PET	VIGIL (interactive games language)	Abacus	3K or 8K expander	$35

XASM

6502 with CP/M	XASM 6500 (Systems Implementation Language)	Succinct Systems	32K min. plus CP/M	$150

XPLO

Ap II, KIM, SYM, AIM	XPLO	6502 Program Exchange	48K, Disk II (Apple II), 20K cass with others	$150 (Ap) includes APEX DOS/$45 (others). $15 user's manual

VIC-20

VIC-20

6502-based home computer manufactured by Commodore
Connects to any TV through RF modulator.
Full-size keyboard includes graphic characters and programmable function keys.

- Graphics:
  - 16 colors
  - Multi-color characters
  - Programmable characters
  - Limited high-resolution
- Sound:
  - 3 Music voices
  - Noise voice
- I/O:
  - Commodore cassette
  - Expansion cartridge
  - Parallel
  - Commodore serial
  - RS-232 (partially implemented)

VIC-20 Memory Map

Address	Function
$00-$FF	Page zero: operating system storage, pointers, floating point accumulators, flags, etc.
$100-$1FF	Microprocessor system stack
$100-$10A	Floating-to-string work area
$200-$2FF	Operating system buffers, tables, vectors, I/O flags, keyboard handling
$300-$3FF	Vectors, tape I/O
$400-$FFF	Expansion RAM
$1000-$1DFF	User BASIC RAM (in unexpanded version)
$1E00-$1FFF	Screen RAM (in unexpanded version)
$2000-$3FFF	8K RAM/ROM expansion
$4000-$5FFF	8K RAM/ROM expansion
$6000-$7FFF	8K RAM/ROM expansion
$8000-$8FFF	Character generator ROM
$9000-$900F	VIC Chip (6560)
$9110-$911F	VIA (6522)
$9400-$95FF	Color RAM (location with RAM at $400)
$9600-$97FF	Color RAM (location without RAM at $400)
$9800-$9BFF	I/O Block 2
$9C00-$9FFF	I/O Block 3
$A000-$BFFF	Expansion ROM
$C000-$DFFF	BASIC ROM
$E000-$FFFF	Kernal ROM--machine independent routines and vectors

Table of Musical Notes

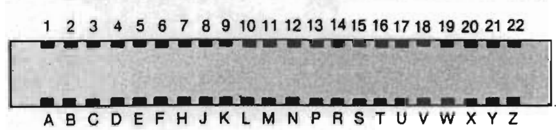
Approx. Note	Value
C	135
C#	143
D	147
D#	151
E	159
F	163
F#	167
G	175
G#	179
A	183
A#	187
B	191
C	195
C#	199
D	201
D#	203
E	207
F	209
F#	212
G	215
G#	217
A	219
A#	221
B	223
C	225
C#	227
D	228
D#	229
E	231
F	232
F#	233
G	235
G#	236
A	237
A#	238
B	239
C	240
C#	241

Poke value into 36874, 36875, 36876, or 36877.
Control volume with POKE 36878,0 to 15.

Expansion/Cartridge Connector

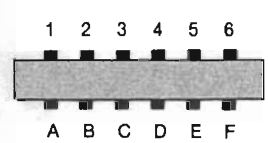
Pin	Function
1	GND
2	CD0
3	CD1
4	CD2
5	CD3
6	CD4
7	CD5
8	CD6
9	CD7
10	BLK1
11	BLK2
12	BLK3
13	BLK5
14	RAM1
15	RAM2
16	RAM3
17	VR/W
18	CR/W
19	IRQ
20	NC
21	+5V
22	GND

Pin	Function
A	GND
B	CA0
C	CA1
D	CA2
E	CA3
F	CA4
H	CA5
J	CA6
K	CA7
L	CA8
M	CA9
N	CA10
P	CA11
R	CA12
S	CA13
T	I/O2
U	I/O3
V	SO2
W	NMI
X	RESET
Y	NC
Z	GND



Cassette

Pin	Function
1	GND
2	+5 V
3	CASSETTE MOTOR
4	CASSETTE READ
5	CASSETTE WRITE
6	CASSETTE SWITCH

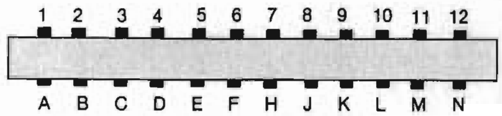


VIC-20

User I/O

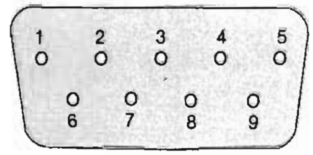
Pin	Function
1	GND
2	+5 V
3	$\overline{RESET}$
4	JOY0
5	JOY1
6	JOY2
7	LIGHT PEN
8	CASSETTE SWITCH
9	SERIAL ATN IN
10	+9V
11	+9V
12	GND

Pin	Function
A	GND
B	CB1
C	PB0
D	PB1
E	PB2
F	PB3
H	PB4
J	PB5
K	PB6
L	PB7
M	CB2
N	GND



Game I/O Port

Pin	Function
1	JOY0
2	JOY1
3	JOY2
4	JOY3
5	POT Y
6	LIGHT PEN
7	+5V
8	GND
9	POT X

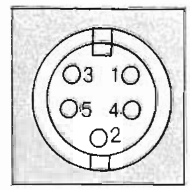


Color Codes

BLACK	0
WHITE	1
RED	2
CYAN	3
PURPLE	4
GREEN	5
BLUE	6
YELLOW	7
ORANGE	8
LT. ORANGE	9
PINK	10
LT. CYAN	11
LT. PURPLE	12
LT. GREEN	13
LT. BLUE	14
LT. YELLOW	15

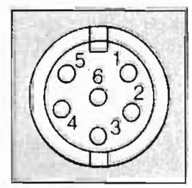
Audio/Video

Pin	Function
1	+5 V, REGULATED
2	GND
3	AUDIO
4	VIDEO LOW
5	VIDEO HIGH



Serial I/O

Pin	Function
1	SERIAL SRQ IN
2	GND
3	SERIAL ATN IN/OUT
4	SERIAL CLK IN/OUT
5	SERIAL DATA IN/OUT
6	$\overline{RESET}$



6560 Video Interface Chip (VIC) Address

Hex	Decimal	Bit	Function
$9000	36864	7	Interlace mode (1on, 0off)
		6-0	Screen origin--horizontal
$9001	36865		Screen origin--vertical
$9002	36866	7	Screen address--bit 9
		6-0	Number of video columns
$9003	36867	7	Part of raster value
		6-1	Number of video rows (0-23)
		0	Character size (8 x 8 0, 8 x 16 1)
$9004	36868		Raster value
$9005	36869	7	Must be 1
		6-4	Screen address--bits 10-12
		3-0	Character memory (value 0-3: ROM, 12-15: RAM)
$9006	36870		Light pen--horizonal
$9007	36871		Light pen--vertical
$9008	36872		Paddle X
$9009	36873		Paddle Y
$900A	36874		Bass voice
		7	Switch
		6-0	Frequency
$900B	36875		Alto voice
		7	Switch
		6-0	Frequency
$900C	36876		Soprano voice
		7	Switch
		6-0	Frequency
$900D	36877		Noise voice
		7	Switch
		6-0	Frequency
$900E	36878	7-4	Auxilliary color
		3-0	Volume
$900F	36879	7-4	Screen color
		3	Reverse mode (normal 1, reversed 0)
		2-0	Border color

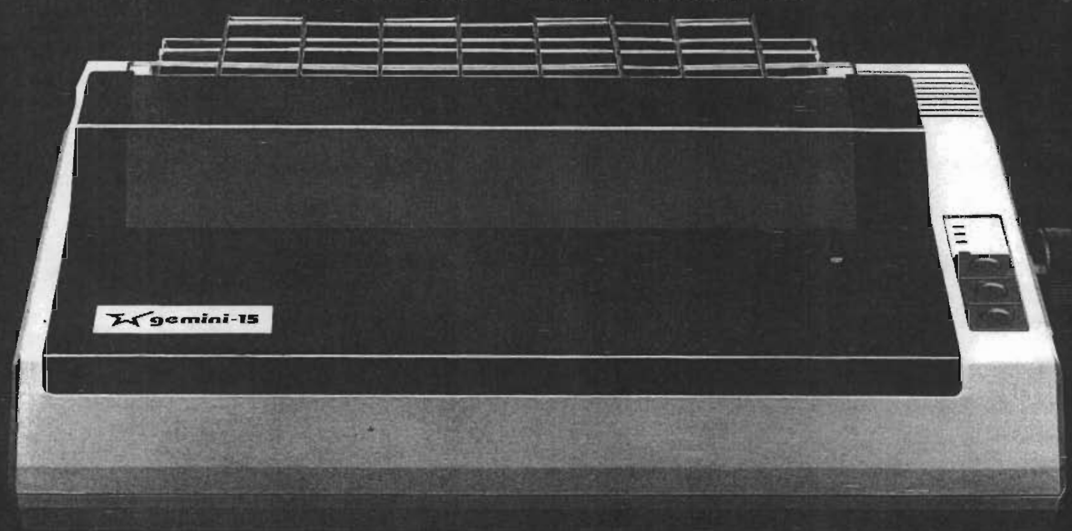
Printers

MICRO™ Information Sheet #2

Manufacturer	Name and Model Number	Printing Method	Speed (CPS)	Number of Columns	Maximum Paper Size	Vertical Spacing	Horizontal Spacing	Character Sets	Graphics	Interface	Price	Comments
Centronics Data Computer Corp. Hudson, NH 03051 603-883-0111	122 Graphics Printer	Dot Matrix, bidirectional logic seeking @10 CPI	120	132	15"	6 lines/in.	5,10 char./in.	7 resident international sets	6 or 8 pin addressable graphics	Parallel or RS232C (Optional)	$995	
Integral Data Systems Milford, NH 03055 603-673-9100	Prism (80 or 132 column)	Dot Matrix	150 or 200	80 or 132 @10CPI	9.5" or 14.875"	6 or 8 lines/in. programmable in 1/48" increments	5 - 16.8 charac./in.	11 character sets	84 x 84 dots/in.	Parallel and RS232C	$1299 or $1499	The printer is modular allowing for upgrades to color, semi-auto sheet feed, auto sheet feed as required
Datasouth Computer Corporation P.O. Box 240947, Charlotte NC 28224	DS 180	Serial Dot Matrix impact	180	132-217	14.875"	6 or 8 lns/in.	5 - 16.5 charac./in.	ASCII, APL, 7 international sets	75 x 72 dot/in. pin addressable	Centroncis Parallel and RS232C	$1595	50 programmable features for forms control, printing and communications; non-volatile format retention
Canon U.S.A. Inc. One Canon Plaza Lake Success, NY 110042	A-1210	Drop on Demand Ink Jet-Bidirect.	40	80	8.5"	1/16 inch	1/12 inch	ASCII	640 dots/line	Centronics Parallel	$795	
Smith-Corona Corp. 65 Locust Ave., New Canaan, CT 06840	TP-1	Daisy Wheel Letter Quality	12	105 or	13.5"	3 - 6 lns/in.	10 or 12 charac./in.	Various print wheels available in traditional and modern faces	No	Parallel or RS-232C	$895	Selectable impression control
Radio Shack Corp. One Tandy Center, Fort Worth, TX 76102	DMP-100	Dot Matrix	50	80	9.5"	6 - 9 lns/in.	5 or 10 charac./in.	Standard ASCII	Block Characters	Parallel or RS-232C	$399	Seven colors: Yellow, blue, red, green, cyan, magenta, and black
Atari Computer Systems 1265 Borregas Ave. Sunnyvale, CA 94086	825	Dot Matrix	40	80-132	9.5"	6 lns/in.	10 - 16.7 charac./in.	Standard ASCII	No	Atari Serial	$799	Does proportional spacing
Apple Computer Inc. 10260 Bandley Dr. Cupertino, CA 95086	Silent Type	Dot Matrix - Thermal	40	80	8.5"	6 lns/in.	12 charac./inch		No	Parallel	$395	
NEC Information Systems 5 Militia Dr., Lexington, MA 02173	Spinwriter	Thimble	35	132	16"	6 lns/in.	10 - 15 charac./in.	Multiple types	No	Centronics Parallel or RS232C	$1475	Does proportional spacing
Williams Laboratories 125 Northview Rd., Ithaca, NY 14850	Bytewriter	Daisy Wheel	12	100	12"	3 - 12 lns/in.	10 - 15 charac./in.	Multiple types	No	Centronics Parallel	$795	Has keyboard; does proportional spacing
Axiom, 1014 Griswold Ave. San Fernando, CA 91340	GP-100	Dot Matrix	30	80	9.5"	6 - 9 lns/in.	12 charac/in.	ASCII	Yes	Parallel or Serial	$389	
Okidata, 111 Gaither Drive Mt. Laurel, NJ 08054	Microline 80	Dot Matrix	80	80 - 132	8.5"	6 - 8 lines/in.	10 charac/in.	ASCII	Yes	Parallel or Serial	$449	
Commodore Business Machines, Inc. 3330 Scott Blvd. Santa Clara, CA 95051	4022	Dot Matrix /Bi-Directional	30	80	9.5"	6 lines/in.	12 charac/in.	ASCII	Pet Character	IEEE	$795	
Epson America, Inc. 3415 Kashiwa St. Torrance, CA 90505	MX-80	Dot Matrix /Bi-directional	80	40 - 132	9.5"	6 - 10 lines/inch	10 - 16.7 charac/inch		120 x 72 Dot Graphics	IEEE, Parallel or Serial	$645	
Qume Corp., 2350 Qume Drive San Jose, CA 95131	Sprint 1140 +	Daisy Wheel	40	132	15"	1/48"	1/120"	Over 100	120x144 dots/sq.in.	RS232C Centronics Parallel IEEE 488	$1786	Does proportional spacing
Diablo Systems, Inc. P.O. Box 5003, Hayward, CA 94545	RO630 API	Daisy Wheel	40	132	15.5"	Adj. to 1/48"	Adj. to 1/120"	240 types	Vector Graphics	Centronics Parallel, RS232C and IEEE 488 by changing cables	$2340	Does word processing functions with escape sequences
C. Itoh Electronics, Inc. 5301 Beethoven St. Los Angeles, Ca 90066	8510A	Dot Matrix	120	80	9.5"	Var. to 1/140"	5 - 17 charac./in.	ASCII and 8 international sets	144 x 160 dots/in.	Parallel or Serial	$795	

Printers

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.

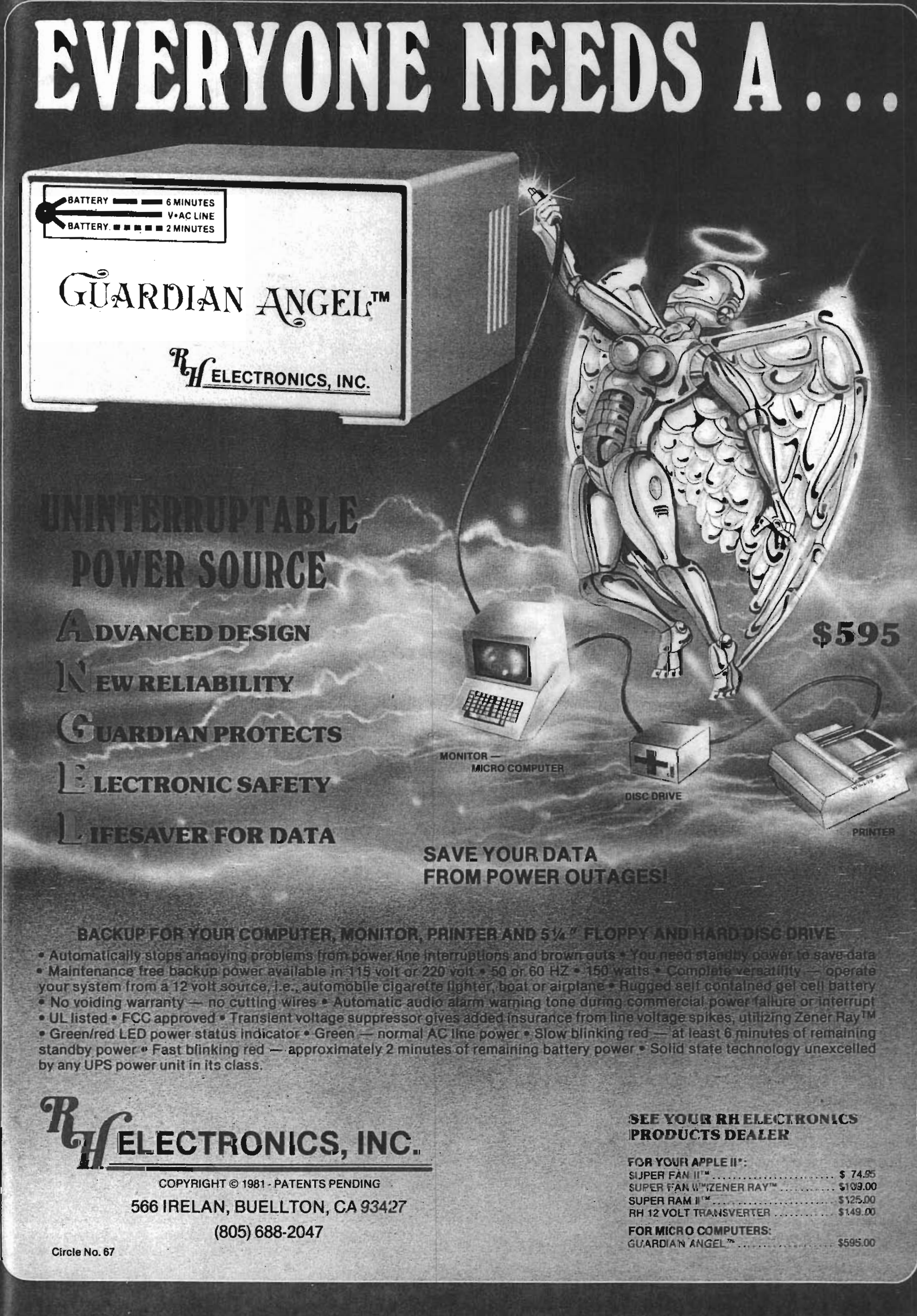
Gemini's flexibility is embodied in its diverse specialized printing capabilities such as super/sub script, underlining, backspacing, double strike mode and emphasized print mode. Another extraordinary standard feature is a 2.3K buffer. An additional 4K is optional. That's twice the memory of leading, 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 over 100 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

Circle No. 66



MICRO™ Interface Clinic

by Ralph Tenny

Some of the fun I have working on the Color Computer is making use of the superb architecture of the basic machine. It is my intent to use this column to help you learn the art of computer interfacing so that you can realize maximum personal use and satisfaction from your machine. I will deal with programming as necessary, but there will be special emphasis on the Radio Shack Color Computer. This machine is one of the most cost-effective designs available for its original purpose of game playing, and it also has great potential for other applications. In many cases, the theory behind my discussions will apply to any microcomputer, but specific details will apply to the Color Computer.

Since I will be talking about computer interfacing, let me define what that means. A computer interface is an electrical circuit that transfers information or electrical energy between the computer and the outside world. This transfer can occur in either direction — from the computer to the outside world, or to the computer from the outside world. I will also use the terms "transducer" and "sensor." A transducer is a device or material that transforms input energy of one form to output energy of another form; a sensor transforms a physical stimulus into an electrical signal.

Fundamentals

(Editor's note: For CoCo users who are beginners with machine language, I recommend Rodney Zak's *Programming the 6809* published by Sybex, or a similar elementary programming book to get started.)

Let me review some microprocessor (μP) fundamentals before I discuss interfacing the Color Computer. The "brain" in your computer is a *m*P that performs nearly one million operations (machine cycles) each second. Any operating computer is reading and executing a list of instructions called a program; making that list is called programming, regardless of the language.

Without a program, the computer is helpless and stupid. However, when you turn on any modern computer like the Color Computer, it already knows how to read the keyboard, display information on the screen, and read programs from the cassette recorder and (perhaps) the disk. It also knows BASIC, and is able to help you learn how to operate it. In other words, the modern home computer already has a lot of program information furnished as part of the package. This column will help you learn to use the existing hardware and software for new purposes and to add more hardware and software to perform new tasks.

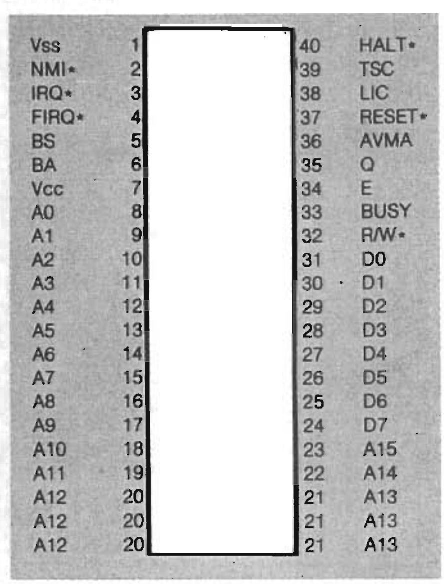
Before I begin discussing interfaces, you need to be aware of some calling conventions used in hardware and logic discussions. First, most logic lines will be assigned names that enable you to discuss them. Where possible, the signature will remind you of the line's function. If a logic line has one logic level (high *or* low) that causes something to happen, that is the *active level.* For example, memory-enable lines usually enable (turn on) the memory for read or write when the line is at logic 0; the line is defined as *active low.* A dual-function line may have a name signifying the function of both levels. There are at least two common ways to signify the active level: the active-low state may have a line over the signature (overbar), or the active-low signature may be designated by an asterisk (MEMENA*).

Figure 1 shows the pinout of the 6809 μP used in CoCo; you can see 16 address lines (A0-A15) and 8 data lines (D0-D7). If the μP sets the address lines to read one of 65536 (2**16) unique memory locations, the memory responds by making available one byte (eight bits) of data on the data lines. The μP continuously performs sequential operations, following instructions in the program. Each byte of an instruction is located at a unique address in the memory of the computer.

Besides the address bus (16 lines) and the data bus (8 lines) already mentioned, the μP has some control lines, two of which are called R/W* (Read/*not* Write; Write is active low) and Q. R/W* signals the memory to furnish data when this line is at logic one (READ is active high), or to receive data when R/W* is at logic zero. Q is a timing signal that tells the memory when the data is valid, and causes the memory to record data presented during a Write cycle.

For now I will bypass how a program is placed in memory. What you need to visualize is an interface to the real world! After all, if the computer can't communicate with the world outside its case, it can't do useful work — not even communicate with its operator! So, besides the memory, which holds instructions and data, the μP must have some sort of external circuit to perform the interface function. Some early μPs had status and flag lines that were connected to package pins on the μP. A status line could be set to +5 volts (logic one) or 0 volts (logic zero) by external signals. The flag lines could be set to logic one or logic zero by the μP, and external circuits would "read" that level and cause some action to happen. For example, a flag line could be used to turn on a light or relay, signalling that the μP had (for example) finished running a program. Most modern μPs (including the 6809) have Programmable Interface Adapters (PIA) that decode the address lines and read

Figure 1: The pinout for the 6809E microprocessor used in the Radio Shack Color Computer. See text for a discussion of pin functions.



or write data on the data bus, translating the μP's blinding speed into fixed voltage levels. PIAs are said to be *memory mapped* — they occupy part of the 65536 memory address locations. Since each address must be unique, memory address space set aside for PIAs and other devices must not be used by memory.

The μP performs its operations so rapidly that any bit of data will be available for less than one microsecond. Figure 2 shows the bus activity for one machine cycle. Assume that six of the eight data lines are connected to a latch or data register as shown in figure 3; if the latch is active (reads the data) when Q drops back to a logic 0 level, then the data is captured by the latch. You should realize also that this latch, like any memory location, must be addressed by the μP only when the data is intended for storage in the latch.

Several numbers appear in figure 2 and represent time in microseconds (μSec), measured from left to right. For example, the time for one cycle of signal E is 1.12 μSec, and Q goes high .251 μSec after E goes low, staying high for .502 μSec. In general, I will present this type of timing diagram using *worst case* times. Later, I will discuss how to derive worst case data. For now, just realize that all integrated circuits have performance variations; i.e., for any number of one kind of IC, some parts will be faster (work better) than others. The manufacturer furnishes data on timing variations, and the worst case data is that performance extreme that will be most likely to cause a circuit design to operate improperly.

Figure 2 represents a snapshot of the precise time interval that data is being sent to the latch in figure 3. How long do you have to capture that data? The data first becomes valid .251 + .028 + .223 (.502) μSec after E goes low. That means that the manufacturer guarantees that a 6809E will take *no longer* than .502 μSec to deliver data to the data bus. Two numbers in figure 2 are followed by (H); this designates *hold time*, or how long the data is available after another signal changes. If you subtract .502 from 1.12, then add .03 μSec hold time, you will derive how long the data is available; the answer is .618 μSec! During the *very next* machine cycle, the data on the data bus will be different and intended for another destination. If you think about it, there are only two things that distinguish the machine cycle of figure 2 from any other machine cycle. First, this is a WRITE cycle, since R/W* is low (W active), so it is sending data out from the μP. Second, each *data destination* must be unique to avoid memory errors and conflicts.

The process of defining a unique destination would be easy if every part that interfaces with the μP had 16 address lines. However, this would require large (and expensive) IC packages to make room for so many pins. Since that isn't feasible, devices called decoders are used. For now, I will explain the function and leave the details until later. One popular decoder has three address input lines and eight mutually exclusive active-low outputs. In effect, if the three most significant address lines (A13-A15) are decoded, the eight outputs will divide the 64K memory address space into eight 8K blocks, with only one block enabled at a time. If you refer to figure 3, the signature DECODE*, working in conjunction with R/W* and Q, produces the latch enable signal STROBE* (shown in proper time rlationship in figure 2).

As a result of the unique combination of signals, the circuit of figure 3 will capture six of the eight bits of the data appearing on the data bus when STROBE* comes true. U2 will hold this data on its output pins until power fails or new data is written into the latch. In other words, the latch output can be considered as a "permanent" copy of the data that was available for only .6 μSec!

If you have a question, or want to suggest a topic for discussion in this column, please write to me at P.O. Box 545, Richardson, TX 75080.

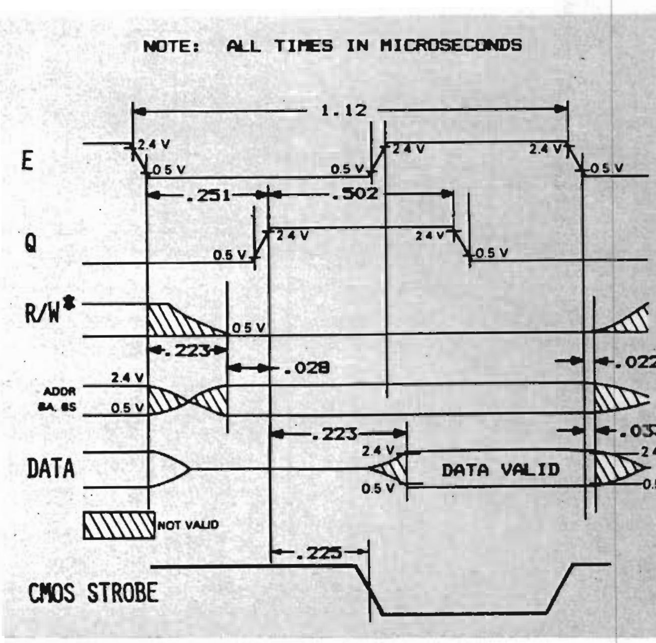
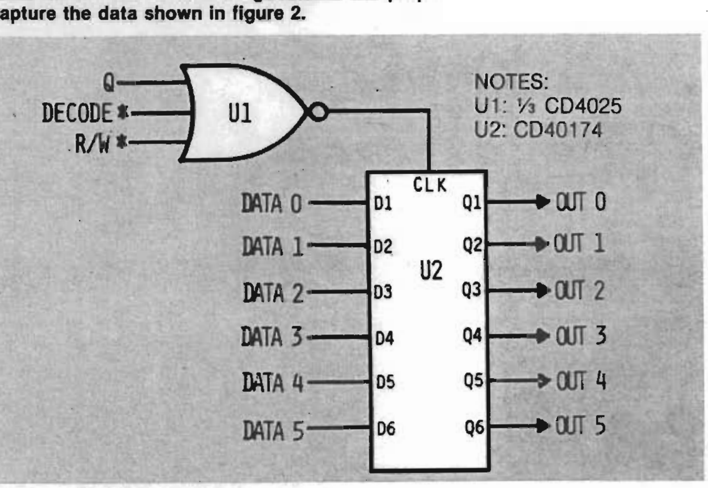


Figure 2 (left): **A basic timing diagram for the 6809E microprocessor, showing the timing relationships during a WRITE cycle. The bottom wave form shows the timing response of the CMOS NOR gate U1 shown in figure 3.**

Figure 3 (below): **One possible schematic diagram to implement a data latch on the 6809E bus. U1 generates the proper strobe to cause U2 to capture the data shown in figure 2.**



MICRO™

Dr. William R. Dial
438 Roslyn Avenue
Akron, OH 44320

6809 Bibliography

113. The Rainbow 2, No. 4 (October, 1982)

Davis, Gary, "Here are Three Assemblers Compared," pg. 30-37.
Includes Cer-Comp, The Micro Works, and Computerware assembler for 6809 systems.

Roslund, Charles J., "Charlie's Machine," pg. 38-45.
A mini-monitor for machine-language use with 6809 systems.

Albrecht, Bob and Firedrake, George, "RND Command is One Secret to Good Games," pg. 46-48.
Tips for the TRS-80 Color Computer game programmer.

Goebel, Kathy, "Line Printer VII: Not So Dumb!," pg. 50-56.
A logo for the Color Computer output.

Ebbert, Jim, "Minicalc Will Handle all Your Projections," pg. 60-62.
A mini-spreadsheet for the Color Computer.

Sherwood, Francis J., "How Much Will It Cost to Buy on Time," pg. 70-73.
A loan amortization program for the 6809-based Color Computer.

Reid, Chris, "Uammalu Provide Lovely Graphics Patterns for 80C," pg. 74.
A graphics program for the TRS-80 Color Computer.

Fraysse, John, "Go Sailing with the Sailor," pg. 76-84.
A game for the 6809-based Color Computer.

Bruck, Bill, "Maintain a Mail List with this Program," pg. 86-90.
A complete mailing list program for the 32K Color Computer.

Lewandowski, Dennis, "Assembly Language Makes Good Graphics Easy," pg. 92-94.
A tutorial for the Color Computer.

Nolan, Bill, "Make Monsters from Silicon — And Use Them Here," pg. 106-109.
Use the Color Computer to generate demons for your fantasy program.

Garrett, Ron, "An Accounts Receivable Program Can Help You," pg. 118-126.
Use the Color Computer to set up a disk-based accounts receivable program to maintain balances on accounts of money owed you.

114. 80 Micro No. 34 (November, 1982)

Shorter, Andrew, "Annual Salary as an Indicator of the Value of Your Savings," pg. 18.
A program for the TRS-80 Color Computer.

Commander, Jack, "CCFORTH," pg. 45-46.
FORTH for the TRS-80 Color Computer.

Norman, Scott L., "Colorterm," pg. 63-64.
A cassette program that converts a 16K or 32K Color Computer into an intelligent terminal.

Barden, William Jr., "The Color Computer on Parade — Part II," pg. 80-87.
A tutorial on graphics and the GET and PUT commands.

Freeman, Jimmy L., "More Color Conversions," pg. 164-170.
Converting TRS-80 Model I programs to Color Computer listings.

Fowler, John D. Jr., "Astrodynamics for Beginners," pg. 205-210.
Use your Color Computer to explore the basic laws of gravitation.

Heusinkveld, John, "Color Assembler," pg. 213-221.
You can speed up your programs with this assembler for the Color Computer 6809 machine language.

Battista, Frederick F., "Show the Score," pg. 272-281.
Put numbers on your 6809-based Color Computer screen while in the graphics mode.

Gilliland, E.O., "Color Computer Pointers," pg. 288-290.
Combining programs on the Color Computer.

Blyn, Steve, "Music Marvel," pg. 308-316.
A program listing to play music on the Color Computer.

Ramella, Richard, "Fun House Adventure," pg. 488-494.
A game for the 6809-based Color Computer.

115. Creative Computing 8, No. 11 (November, 1982)

Norman, Scott L., "Alternative Languages for the Color Computer," pg. 124-138.
Assembly language for the Color Computer can add speed to your programs.

Wilkins, Dennis, "An Editor-Assembler for the Color Computer," pg. 142-147.
An inexpensive way to obtain some editing features for 6809 machine-language programs.

Ahl, David H., "Build a Paddle Control for Your TRS-80 Color Computer," pg. 242-244.
A hardware article to provide paddles at a modest cost.

116. Color Computer News, No. 13 (October, 1982)

Steiner, John, "Morse Code Instruction: Part 1," pg. 19-26.
Programmed CW instruction for the TRS-80 Color Computer.

Kilpatrick, Roger, "Color Data File," pg. 27-35.
A program for storing data on the Color Computer.

Ashley, Peter, "Film Development," pg. 36-42.
A 6809-based Color Computer program to handle the various factors involved in photo developing.

Giovanoni, Richard, "Logchart," pg. 43-44.
A Color Computer program to draw a sheet of 2 x 3-cycle log paper on the screen and adapt it to plot an aircraft log.

Greb, Jack, "Disk 'DIR'/Ectory," pg. 46-49.
A Color Computer program to assist in reading long directory files.

Phelps, Andrew, "Comment Corner," pg. 51-54.
Notes on the Color Computer error messages.

Hogg, Frank, "64K Korner," pg. 55-56.
Notes on the operating systems for the 6809-based Color Computer.

Hass, E.J., "Neatprint," pg. 62-64.
A Color Computer program to assist in printing out listings.

Brown, Jim, "Memory Test," pg. 68-70.
A utility for the 6809-based Color Computer.

White, Richard A., "Grades: A Computerized Gradebook," pg. 73-78.
A grade-book program for the teacher using a Color Computer.

117. Personal Computer World 5, No. 11 (November, 1982)

Staff, "In Store," pg. 184-191.
A tabulation of the specifications and main features of 113 different microcomputer systems, including 6809's.

118. Byte 7, No. 11 (November, 1982)

Barden, William Jr., "Inexpensive Transducers for the TRS-80."
Part 2 on monitoring real-world quantities covers switch-sensing techniques for the TRS-80 Color Computer.

MICRO

Next Month in MICRO

April Communications Feature: Computers are reshaping trends in the communications field. Find out what's happening by reading the April issue of MICRO. You'll also discover information on system-specific communications and bulletin boards.

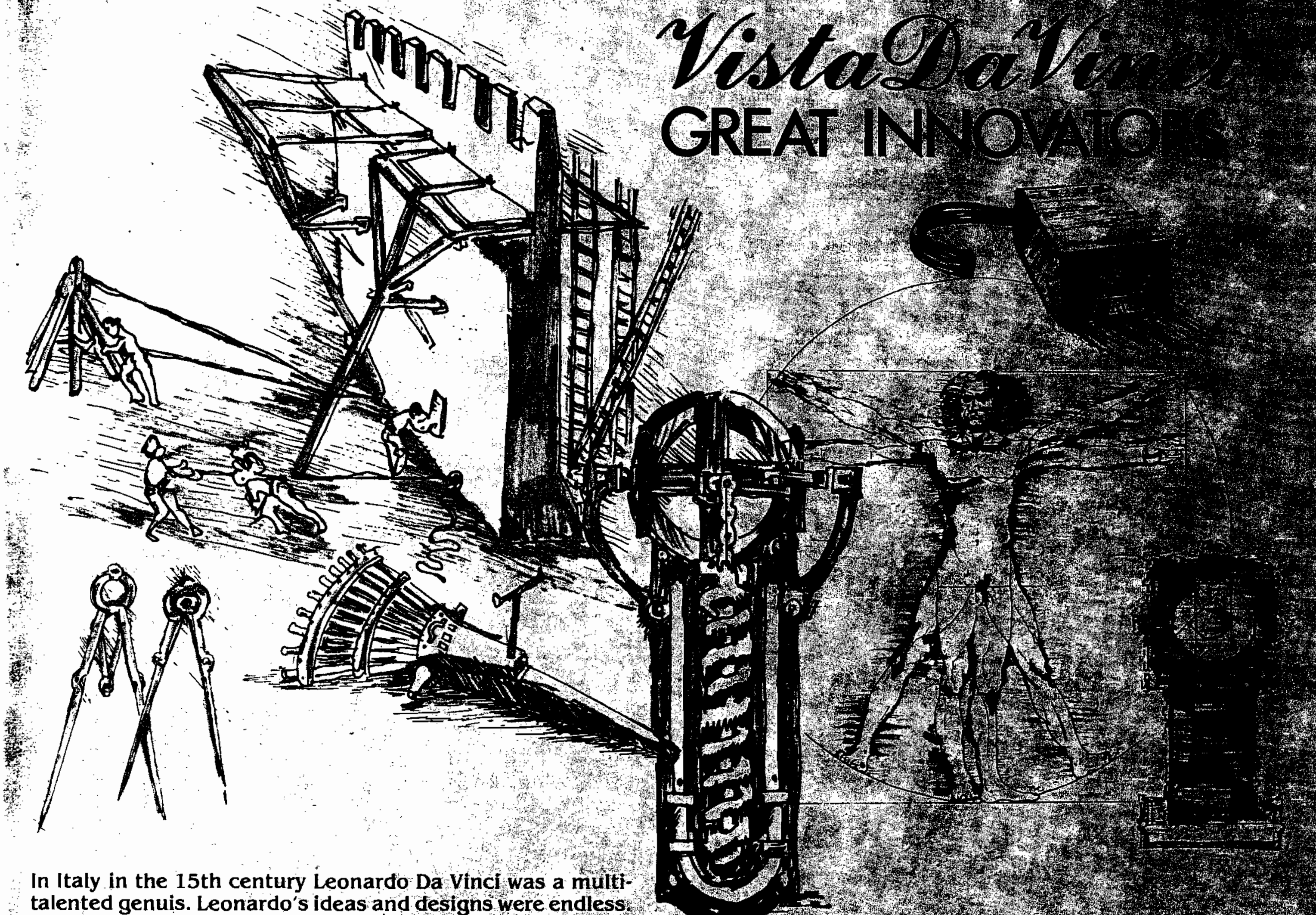
Next Month's **Learning Center** features *Master* for Commodore and Apple — a simple guessing game for one or two players.

Plus: Apple LETTERMASK
Interfacing the Atari Joystick to the Color Computer
Mutual Funds on the OSI
Bond Quotations on the Apple

Advertiser's Index

ABC Data Products . . . 54
Acorn Software Systems . . . 62
Alternative Energy Products . . . 83
Animation Graphics . . . 40
Anthro-Digital Software . . . 27
AppleTree Electronics . . . 41
Ark Computing . . . 92
Armidillo Software . . . 13
Artsci,Inc. . . . IFC
Chameleon Computing . . . 74
Compu $ense . . . 13,18,26,35
CompuTech . . . 66
The Computerist, Inc. . . . 64
Computer Case Co. . . . 7
Computer Mail Order . . . 68-69
Computer Science Engineering . . . 37
D&N Micro Products . . . 9
Datamost, Inc. . . . 22, 110
Digital Acoustics . . . 12
Execom, Inc. . . . 87
Gibson Laboratories . . . 98
Gimix, Inc. . . . 1
Gloucester Computer Bus Co. . . . 88
Hudson Digital Electronics . . . 84
IJG Company, Inc. . . . 76
Intec Peripherals Corp. . . . 11
Interesting Software . . . 23
John Bell Engineering . . . 30
L Com . . . 95
Leading Edge . . . BC
Logical Devices . . . 95
Lyco Computing . . . 56
Meta-Lab . . . 87
Microhouse . . . 4
Micro Motion . . . 54
Micro Peripherals, Inc. . . . 44
Microtech . . . 75
Micro-Ware Distributing . . . 81
Modular Mining Systems . . . 37
Modular Systems . . . 37
Optimized Systems . . . 60
Penguin Software . . . 6
Performance Micro Products . . . 35
Perry Peripherals . . . 34
RH Electronics . . . 17,107
Rock Roy . . . 3
Scientific Software . . . 83
SGC . . . 99
SJB Distributors . . . 71
Skyles Electric Works . . . 63
Smartware . . . 62
Sof/Sys, Inc. . . . 23,55
Sorrento Valley Assoc. . . . 93
Southwestern Data . . . 11,96
Star Micronics . . . 106
Strobe, Inc. . . . 91
Time Trend Software . . . 98
Versa Computing . . . 89
Vista Computing . . . IBC
XPS, Inc. . . . 83
Zytrex . . . 13
MICRO Advertising
Mastering Your VIC-20 . . . 46
MICRO on the OSI . . . 38
What's Where in the Apple . . . 80

MICRO is not responsible for claims made by its advertisers. Any complaint should be submitted directly to the advertiser. Please also send written notification to MICRO.



VistaDaVinci
GREAT INNOVATORS

In Italy in the 15th century Leonardo Da Vinci was a multi-talented genuis. Leonardo's ideas and designs were endless. He was always creating new devices to solve the world's problems. Pictured here are many studies for military arms and armor and battlements.

Today Vista is developing many high technology computer peripherals for use with IBM™ and Apple Computers.

Introducing Solo, a new cost effective solution to Apple Disk storage offering all the capabilities of the standard Apple Disk II at a fraction of the cost.

If your needs do not demand the high performance of our Duet, Quartet, or V-Series Maxidrives, then Solo will provide you with what you need at the lowest possible cost.

Vista Solo™

The Vista Solo incorporates the proven reliability of the Shugart™ 5¼ Drive. To date Shugart has over one million units in operation throughout the world.

Solo is also available as the Solo Plus, which comes with the Solo Controller. The pair give you total compatibility with Apple hardware and software.

- Shugart Drive
- Totally Compatible
- Cost Effective
- 35 Track Compatible Drive
- 143K Storage
- Full Vista 120 Day Warranty

Solo (Drive only) #3101
Solo Plus (w/Controller) #3111

Contact Your Local Vista Dealer or Call our Vista Hotlines.

Vista COMPUTER COMPANY, INC.
1317 East Edinger / Santa Ana, CA 92705
(714) 953-0523 / (800) [illegible]017

Distributors

California: Group III Electronics [illegible]
(213) 973-7844 [illegible] 731-2307 [illegible]

California: Micro D [illegible]
(213) [illegible]

[illegible]

™Shugart is a registered trademark of Shugart Associates.
™IBM is a registered trademark of International Business Machines.
™Apple II is a registered trademark of Apple Computer Co.