NO. 66 GRAPHICS
U.S. Edition: $2.50
International Edition: $3.00

Complete Programs Included

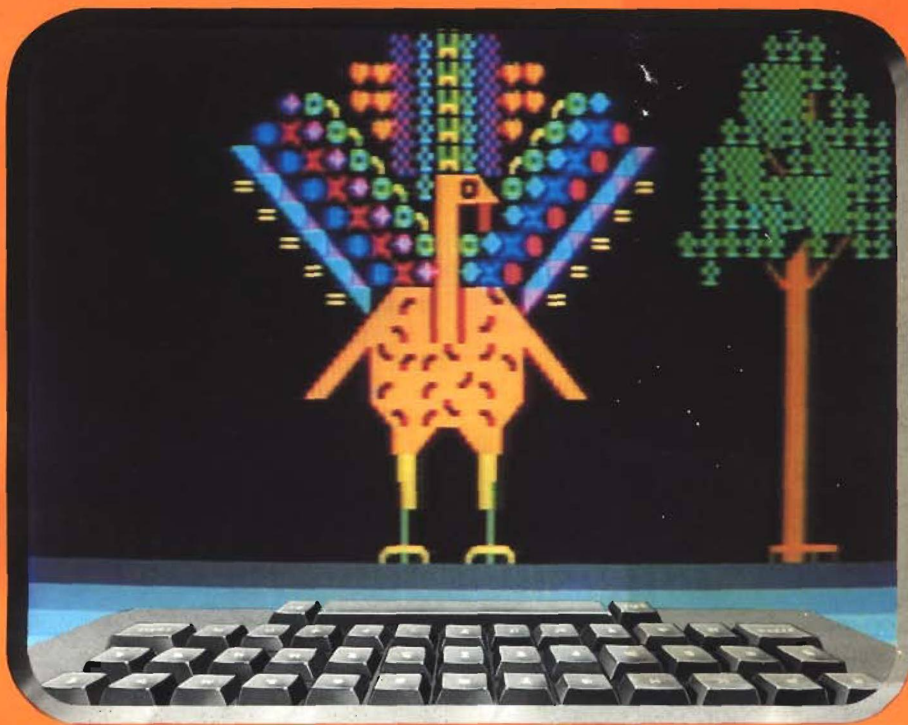
NOV 83

MICRO™

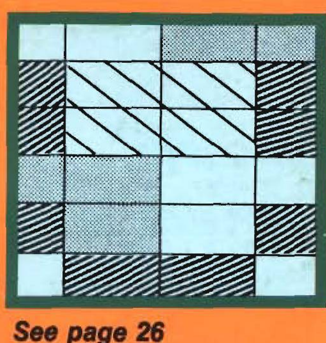
Advancing Computer Knowledge

EXPLORING CHARACTER GRAPHICS

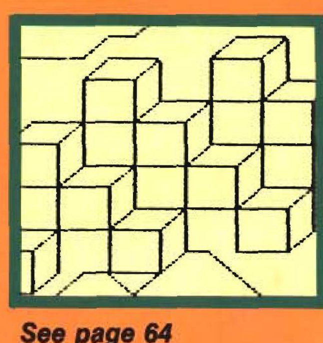
- Screen Editor
- Printer Dump
- Draw Package
- Template Program



Complete Programs for Apple, Atari Commodore and TRS-80



See page 26



See page 64

See page 90

Apple Bar Graph Generator

Atari Mode 10 in Mode 0

Radio Shack Introduces OS-9

SCRIPT 64™ Makes Your World Letter Perfect

SCRIPT 64 moves you into the exclusive neighborhood of sophisticated word processing for an amazing $99.95.

In addition to all the standard text editing features most higher priced systems offer, SCRIPT 64 also gives you an 80 column format and a built-in spelling dictionary...all in one program for the Commodore 64™.

To make your world letter perfect, ask your dealer for a demonstration of SCRIPT 64 today.

Developed by:

RTC

Richvale Telecommunications, LTD.

A Product of:

Blue Sky Software
Ashland Office Center
Evesham & Alpha Avenues
Voorhees, NJ 08043
609-795-4025

Distributed by:

Warehouse 1, Inc.	Computer Marketing Services, Inc.	DES-Data Equipment Supply
Eastern U.S. 800-253-5330	800-222-0585	213-923-9361
Western 800-255-0056		

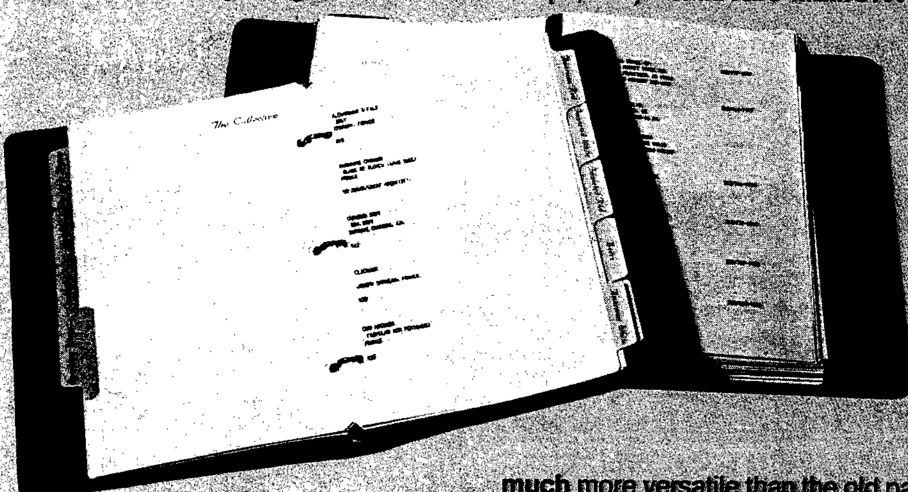
Also available for the IBM-PC.

MAGIC MEMORY

Will Remember Everything For You

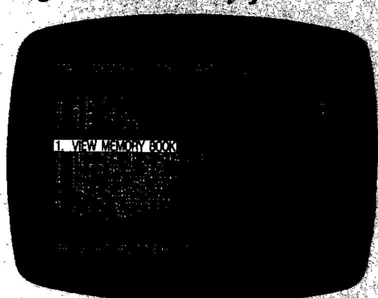
(If you want it to)

Imagine a system that would record all the wonderous, valuable information you have assimilated onto a single tiny disk. (No more scattered bits of paper, business cards, etc.) Imagine the same system giving you a typed sheet you could put into a notebook or print out for a party and instantly change, or add to, at a moments notice. Imagine cross-referencing to suit both your business needs and personal desires so that all your data was organized into one little black book! On top of all this — imagine having **fun** putting it together.

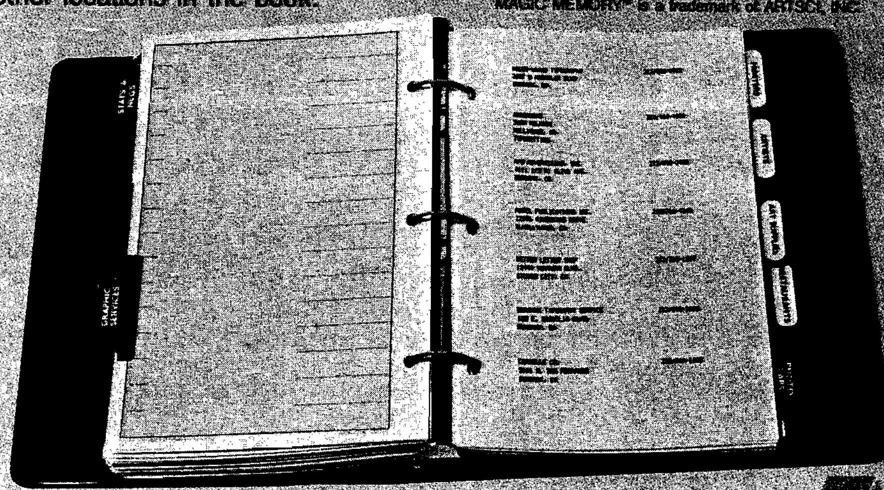


"Imagine" no more. ARTSCI, INC. proudly announces **MAGIC MEMORY™**, the complete advanced database system. **MAGIC MEMORY** is made to simplify the information storage process.

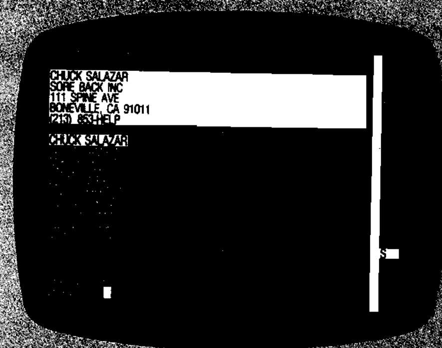
The typical first-time computer user has no *human* experience that will help him to relate to a *computer's* methods of handling information. Therefore, his learning ability is hampered and remains so for many hours of use, UNLESS the computer has been designed to run simply.



MAGIC MEMORY™ is built for the computer rookie. Everyone can relate to **MAGIC MEMORY™** because its form is familiar. It looks like an address book but its not. Its *more*. Like the address book, **MAGIC MEMORY™** presents an A thru Z index tabulation on the right edge of the video display. The user simply selects a tab and the *book* is opened to the proper page(s). A second set of tabs are available that can be labeled by the user (i.e.: companies one deals with, birthdays, lists, wines, shops, etc.) Yet **MAGIC MEMORY™** is much more versatile than the old paper address book. This system can instantly add or delete information, sort alphabetically, and transfer data to other locations in the book.



One of the features in the print section of **MAGIC MEMORY™** is the ability to print out any or all of the *address* book, or any of its special sections.



MAGIC MEMORY'S data presentation screen will instantly adapt to any type of information you may want to store. It is only limited to your imagination and your needs.

MAGIC MEMORY™ is designed to operate on an APPLE *IIe* and still remain totally compatible with APPLE II. The system will operate in *40 columns* or 80 columns. You may also use the 70-column display that requires no additional hardware.

MAGIC MEMORY™ retails for $99.95. Call 213/985-5763. Use your imagination to outwit your organizational woes.

213/985-5763
5547 Satsuma Avenue
North Hollywood, Ca 91601

artsci

MICRO™ Editorial

Frustrations, Features and Futures

The Frustrations

I find the personal computer absolutely fantastic, facinating, and, very frustrating. The new "under $200 computer" recently purchased for my kids has many wonderful built-in features. Unfortunately, there is almost no support for using and/or understanding these capabilities. For example, this new computer has many great video features that were not available on earlier systems. These include more colors, more graphic characters, user-defineable characters, and so forth. BUT, the computer does NOT have a built-in graphics editor. The built-in text editor is designed for entering BASIC programs, and has severe limitations when used for graphics. And, if the minimal text editor is used to generating graphic displays, there is no built-in provision to SAVE the screen to tape or disk, or to LOAD a display screen from tape or disk. And, there is no built-in way to convert the screen display into DATA statements or PRINT strings that can be used directly by a BASIC program. And, the editor does not provide any programmable character, line drawing, or higher-level graphic support. My kids and I find this very frustrating. Having capabilities which are difficult or impossible to use is almost worse than not having the capabilities at all!

One obvious solution is to purchase a software package designed to do graphics. That is find if we want to do enough graphics to justify the cost. And, if we can find a software package which does what we want. And, if we do not care to know HOW the microcomputer operates — since most packages do not provide listings. And, if we never need to modify the software and/or add features.

The Features

Some of MICRO's readers are inexperienced with microcomputers and may not even be aware of the possibilities that they are missing. Other readers are advanced experts who could write software to solve many of the problems. In any case, writing programs to provide the basic service which the microcomputer needs takes time and effort. The MICRO solution will be to take one significant problem area each month and provide solutions. Not just talk, but computer programs that solve the problems. While there will be detailed discussions of the problems, alternatives, "packaged" solutions, and so forth, the heart of each issue will be one or more complete, ready-to-run programs that you can use with your micro. Each program will be written to run on as many of the microcomputers that MICRO regularly covers as possible.

In this issue we present five graphic programs. Each program is listed in its entirety, conversions are NOT "left as an exercise for the reader"! These include the following: Graphic Screen Editor, Programmable Character Generator, Drawing Program, Template Program, and Graphic Screen/Printer Dump.

The Future

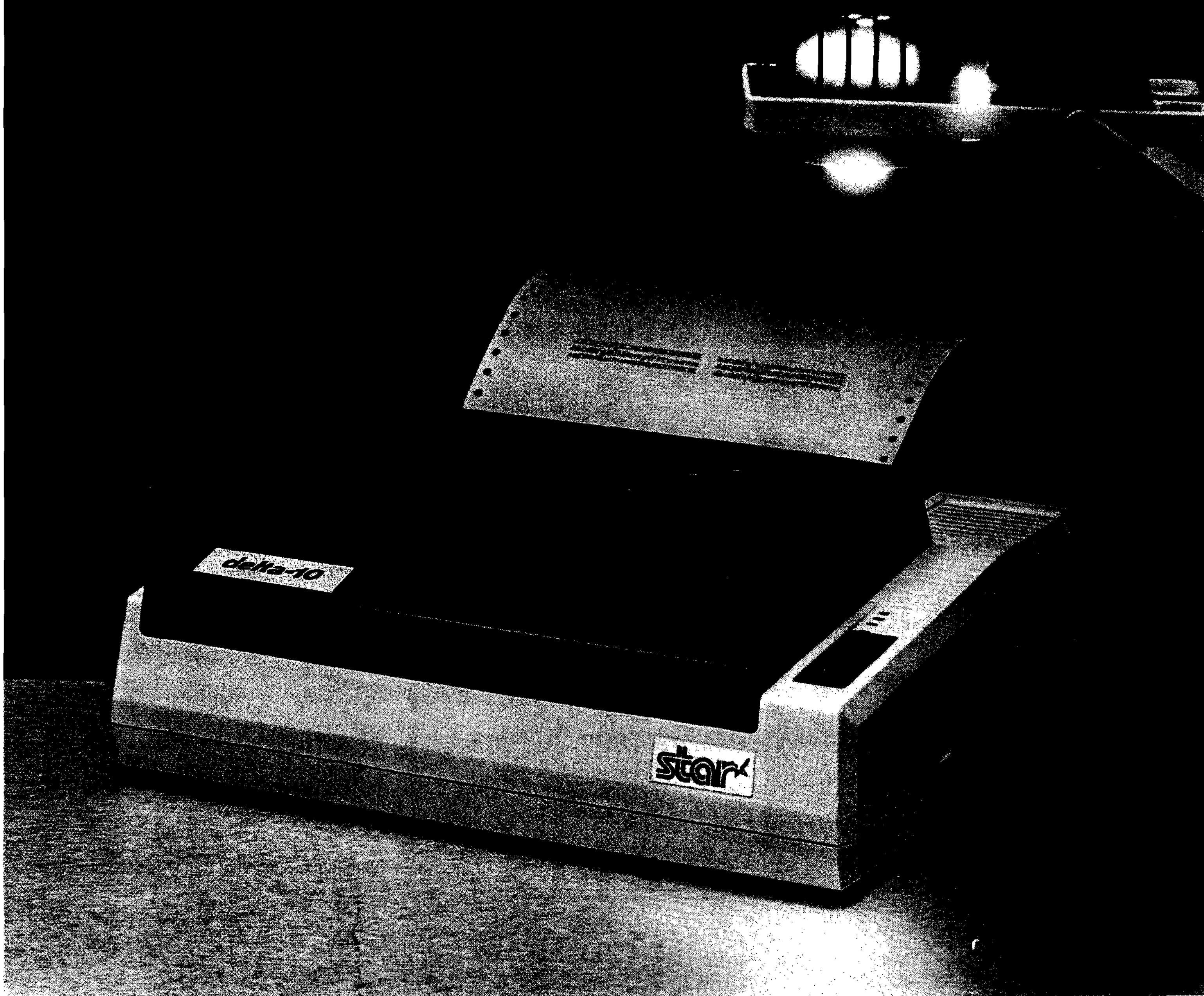
MICRO will continue its general coverage of the Apple, Atari, Color Computer, Commodore computers and other interesting personal computers. Each system will have its own special section with microcomputer specific information such as articles, reviews, catalogs, columns, plus the machine specific program listings and information that relate to that issue's feature topic.

Topics in coming months will include: Electronic Spread Sheets plus MicroCalc, Local and Remote Communications, Personal Data Bases. With each issue containing programs that you can use in your system, MICRO will become an even more important asset in your microcomputer world.

Robert M. Tripp

President/Editor-in-Chief

MICRO™



delta-10

MICRO™

Advancing Computer Knowledge

MICRO
P.O. Box 6502,
Amherst, NH 03031
(603) 889-4330

MICRO is published monthly by: MICRO, Amherst, NH 03031. 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, 10 Northern Blvd., P.O. Box 6502, Amherst, NH 03031, or call (603) 889-4330, Telex: 955329 TLX SRVC, 800-227-1617. *Subscription rates (per year)*: U.S. $24.00, $42.00 / 2 yr. 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.

Graphics

Commodore

Exploring Character Graphics
begins on page 11...

Color Computer

Atari

Apple

DEPARTMENTS

Letterbox

More OSI Feedback

Dear Editor:

I am writing you today to decry your dropping of OSI, and other low-end, "old technology" microcomputers such as the AIM-SYM-KIM. Of course, you cannot devote much space to them, and, as you have pointed out, you don't receive many articles about them, either. But the only other "national" source one might have for OSI information (besides the newsletters, which do not fit in the same category) would be BYTE — except BYTE specifically does not include that machine.

I agree that OSIO and PEEK(65) are the most valuable resources for certain types of nitty-gritty information about the pushing of electrons through the OSI. But if you received the occasional article which was well done, would you be averse to using it? This month's issue (August) has an exciting terminal program I am dying to try out to replace my current (commercial) program. In the last few months, you have been putting out quite a few articles, which have been a welcome sight. Were you only "flushing the buffer" before closing your files, so to speak?

We have OSI's in our labs and I have an OSI at home; the reason I don't show up on your subscriber list is because my institution has a subscription to support the OSI's and other micro's here. As you change your editorial focus, I hope we don't grow so far apart that we don't need each other anymore. MICRO has been a good friend and a source of comfort to me, and I'd hate to break that up.

Good luck at any rate!

Ted Morris
6306 Kincaid Road
Cincinnati, OH 45213

Dear Editor:

I was extremely dismayed by your editorial decision to eliminate all coverage of Aim/Sym/Kim computers. Although the commercial market for ASK products has essentially died, I feel that you have betrayed those of us "Single Boarders" who helped to create this magazine. I remember when MICRO was a badly printed amateurish periodical. I subscribed to it sight unseen because I wanted to support what I knew was a small, yet extremely interesting segment of the computer market. I was very pleased that MICRO went on to become a commercial success. To help insure your success and growth I took advantage of your offer to buy an extended subscription to MICRO. My subscription is scheduled to expire in June of 1988. However, since you have chosen to abandon coverage of single board computers, I feel that MICRO and I no longer have enough common ground to be worth continuing my subscription.

The market for ASK computers has completely shriveled to almost nothing, but I still use my Sym, and plan to continue using it for years to come. In fact, I recently purchased several expansion boards for it at "close out" prices. User club newsletters are great, but they are basically run strictly for fun. MICRO had been the only truly professional magazine that we ASK fanatics could read and enjoy. We gave you your start, but now you refuse to even acknowledge our existance. I wish you well as a commercial enterprise. Your magazine is very well done, but I won't be reading it anymore.

J. R. Casey Bralla
Box 162, Rte 9
Asheboro, NC 27203

Computer Camps

Dear Editor:

My wife and I had the same problem as one of your letter writers in your September issue: How do you find the right computer camp for your child or yourself? There was no available guide or book on computer camps, so we thought we'd write one. How many computer camps could there be anyway?

One year later we know how many computer camps there are: at least 300 (!), and we have them all described, listed, and evaluated in our new book *Computer Camps!*, which is being published by Bobbs-Merrill this spring.

Our book is the most comprehensive guide ever published about computer camps, and lists camps in 40 states, seven foreign countries, and on three ocean cruise liners. We found camps for kids, camps for adults, camps for teachers, and camps for the whole family.

If your readers can't find the book in the stores yet, contact us at the below address and we'll try to help them!

Mike and Laura Benton
THE COMPUTER CAMP PROJECT
P.O. Box 50028
Austin, TX 78763

TI 99/4A Information

Dear Editor:

Thank you for John R. Raines' article, "Dvorak Keyboard for Your Computer," in the August 1983 issue of MICRO. As a long-time user of the DSK ("Dvorak Simplified Keyboard"), I am glad to see renewed interest in a keyboard that has over and over been shown to be significantly superior to the outmoded "QWERTY" keyboard. Thank you for publishing the article!

Along with my commendation, I am afraid I must also include a minor complaint with reference to the August 1983 issue of MICRO. I was very disappointed to read the "Peripherals Catalog for Texas Instruments." Here is what your article said:

There are no second-source Original Equipment Manufacturers of hardware for the TI 99/4A. All hardware available is produced by Texas Instruments. As the enclosed full-page advertisements indicate, there are many others other than Texas Instruments who are supplying hardware for the TI-99/4A: Percom Data (advertising their disk drive for the TI-99/4A), A J International (advertising their RS232 interface for the TI-99/4A), CompuTech Distributing (advertising the Myarc Winchester disk and controller system for the TI-99/4A), and Foundation (advertising their 128K RAM memory card for the TI-99/4A.

Some others supplying hardware for the TI-99/4A are the following: Ultracomp (Ultra-Ram Modules and Ultra-Smart RAM Modules), A/D Electronics (First Ade Control Card, including 8 analog input channels, 8 digital input channels, 8 digital output channels, and real time clock, First Ade Interface, etc.), Tachyon Systems (32K RAM Memory), Morning Star Software (CP/M Processor, with CBASIC Interpreter also available), ROMOX (Widgit Cartridge Expander Board and EPROM Cartridges), Intellitec Computer Systems (32K Memory, RS232), Doryt Systems, Inc. (Paraprint 18A interface), etc.

Many disk drives, of course, are compatible with the TI-99/4A — such as Siemans, MPI, and Shugart — and the ones sold with TI's name on them are in face made by such firms as Siemans, MPI, and Shugart. Much other hardware — although not made perhaps specifically for the TI-99/4A — is compatible with the TI-99/4A (e.g., the Signalman Mark III modem).

Barry A. Traver
552 Seville Street
Philadelphia, PA 19128

MICRO

Updates and Microbes

Who Wrote That Letter?

We regret that Mr. George W. Sherouse's name was omitted from the letter entitled "A New Look at a Full Byte" in our July Letterbox. We appreciate Mr. Sherouse's comments and want him to receive full credit for his ideas.

September Typo

"Signed Binary Multiplication with the MC 6809" by T. J. Wagner and G. J. Lipovski (64:111) contained two misprints. In the last column on the right 6_7*2^8 should read b_7*2^8, and the fourth equation down should end with (2), not (1).

Less Than, Not Greater Than

In Dan Weston's "Hi-Res Characters for Logo" (64:50), listing 4 contained an error. The correct line should read:

MOVECURSOR
TEST COLUMN<39 (not>)

MICRO

Have you got a point of view that you'd like to share? Then write to us. Your letters play an important part in helping us make MICRO the magazine that you want to read. We always welcome reader input. Please send your comments, insights, ideas, or bugs to Letterbox, MICRO, P.O. Box 6502, Amherst, NH 03031.

Outdated Computers Pose Threat To National Security

Newsweek reports that the Pentagon is plagued by outdated, inefficient and often incompatible computer systems that pose a potential threat to national security. Antiquated "conventional" computers currently in use by the Defense Department to handle logistical chores are wasting millions of tax dollars annually and fail to provide a centralized, automated inventory of spare parts for the nation's military systems.

According to a recent report by President Reagan's Private Sector Survey on Cost Control, taxpayers could save up to $9.1 billion over three years through improved Pentagon automation.

New Computer Series Premieres

"Educational Computing Profile," a new monthly television series updating parents and educators on the quality of electronic products, services and activities, premieres this month on public television.

The September edition of the series compares three popular computers, evaluates two software packages and features a round-table discussion on computer literacy with Irwin Landau, editor of "Consumer Reports" and Rowan Wakefiled, editor of "American Family" newsletter. Another segment of the half-hour program reports on new trends in electronic learning.

"Education Computng Profile" is produced by KET, Kentucky Education Television. Educational Products Information Exchange (EPIE) is responsible for the content of the shows.

Computer Crime

In the next decade, more bank heists will be committed by computer, terrorists will make computers their polictical targets, and incidents similar to the tampering of patient computer records at New York's Sloan-Kettering Cance Center will become more frequent.

These are the predictions of Sanford Sherizen, an instructor in the College of Criminal Justice at Northeastern University, whose firm, Data Security Systems, Inc., helps companies deal with computer theft — "the white collar crime of the 80s," he said.

In his course on white collar crime at Northeastern he told his students that although high tech bandits and their escapades are a widespread and growing problem for many industries, few administrators are taking this costly threat seriously.

In his work, Sherizen develops a security assessment for companies. He often recommends a streamlining of the computer password system and the coding of data to prevent theft.

UPI Added to NewsNet System

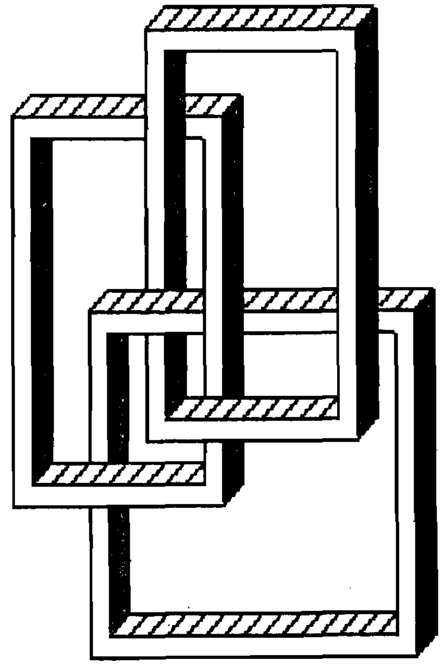
All national and international news from United Press International is now available to users of NewsNet, the nation's leading electronic newsletter publishing specialists.

To use NewsFlash, subscribers create a "keyword profile" of up to 10 words or phrases that reflect their areas of interest.

NewsNet, which began operations in April 1982, offers full text retrieval at 300 and 1200 BAUD of more than 125 business newsletters. Anyone with a microcomputer, terminal, or word processor, equipped with a modem, can use NewsNet. NewsNet is wholly owned by Independent Publications Inc., of Bryn Mawr, PA. For further information, contact NewsNet toll-free at 800-345-1301. (In PA call 215-527-8030.)

MICRO

EXPLORING CHARACTER GRAPHICS



by Robert M. Tripp, Phil Daley, and Loren Wright

Exploring Character Graphics

One of the great features of the modern microcomputer system is the use of a video display as the primary output device. It was not that many years ago when the major output device of even very expensive computer systems was the Teletype®. This slow, noisy, expensive device could provide only a limited hard-copy output, consisting of uppercase letters, numbers, and punctuation characters in black and white. What a difference from the capabilities of the modern microcomputer with its full color, high-speed video display. With all the spectacular things that computers can do using high-resolution graphics, sprites, players and missles, it is easy to underestimate the power and usefulness of character graphics. Of course you *use* character graphics all of the time. All of your programming, word processing, and electronic spread sheet work use the basic alpha-numeric characters of the computer. Have you ever seriously explored the other capabilities of these character graphics on *your* microcomputer?

This issue of MICRO focuses on **Exploring Character Graphics**. We are not just going to talk about the topic; included in this issue are a number of complete computer programs — tools for using and exploring character graphics. The programs for each specific microcomputer — Apple, Atari, CoCo, Commodore and VIC — are found in the system specific sections of the magazine. The 'ready-to-run' programs include:

Screen Editor with Save/Load/Print
Programmable Character Generator
Line Drawing Program
Template Generating/Drawing Program

Due to differences in their basic character graphic capabilities, not all programs are possible for all of the microcomputers.

The Importance and Evolution of Character Graphics

What is a Character?

Webster's defines *character* as "a graphic symbol (as a hieroglyph or alphabet letter) used in writing or printing". There are many other definitions of *character*, but we can all understand this basic concept. The most common characters we encounter are the letters that make up the alphabet. While these letter characters are primarily used for communication with other human beings, there are special computers, called *optical character readers*, that can also *read* our alphabet. Other types of characters which you might consider are:

Magnetic characters on bank checks
Bar Code characters on commercial products (such as on the cover of this magazine)
Symbols for maps, electronics, choreography, architecture, and others

The Evolution of Computer Characters

Computers do not use characters internally. All of the information in the computer is in the form of electronic bits — ones and zeros (which we will refer to here as internal codes, or just codes). Whether or not a particular code represents a character depends on the device that is interpreting these codes. The code whose bit pattern is

□ ■ ■ ■ ■ □ ■ □ HEX 7A DEC 122

could be interpreted as the character **'z'** on an ASCII terminal with upper and lower case; as **'Z'** on an ASCII terminal with upper case only; as **':'** on an Apple II; as a **'graphic character'** on a Commodore 64 display in one particular mode or a **'check mark character'** in another mode; a **'diamond'** if a PRINT CHR$(122) is performed, and so on. The only time the computer is concerned with characters (as opposed to internal codes) is when it is dealing with input or output. And, it usually takes a lot of hardware and processing to turn an external character into an internal code within the computer, or to convert an internal code into a character pattern for output to some device.

In the earliest computers, there was no concept of *character* at all. Information was entered by changing external wires on *patch panels*. The next step was to control the input of information into the computer via a set of switches. It really was **All ones and zeros**. Information was output via a row of lights. You would really be stretching it a bit to call:

□ ■ □ □ □ ■ □ ■

a character. A major conceptual step was achieved as computers were hooked up to Teletype machines, permitting the operator to enter information in standard characters for the first time. Granted, the Teletype was limited to upper case letters, numbers, punctuation and a few control codes — but it was a giant step forward. Even if the string of characters being input was meaningless to the operator — at least in a language sense — it was certainly much easier to enter:

A9 14

than to set switches to read:

■ □ ■ □ ■ □ □ ■
□ □ □ ■ □ ■ □ □

The IBM punch card is another example of the relationship between the human oriented character and the computer's requirement for binary data. The correspondence between the alphabetic characters and their numeric codes is very apparent on the IBM card. On the top line of the card the keypunch machine types the familiar typewriter characters, but underneath, the little rectangular holes are arranged to form a unique binary code for each different character. The operator enters information into the keypunch machine in alphanumeric characters; the computer card reader reads the binary holes.

A Broader Definition

Besides the alphabetic, numeric, and punctuation characters, the computer understands and uses other codes that have **no** character associated with them. These include the typewriter-like functions — carriage return, line feed, and tab. Other functions such as cursor movement, inserting and deleting characters and lines, home, clear screen and clear-to-end-of-line and -page make sense on a video terminal. Additional functions such as stop list, break, changing colors, or reversing the image depend on the particular brand of microcomputer.

Standard Codes

Early in computer evolution, many different coding systems were devised. Baudot was a system used with teletypes and designed for communication, not computers. Many mainframe manufacturers had their own unique internal coding. EBCDIC (Expanded Binary Coded Decimal Interchange Code), for example, is a coding system used on large IBM computers, including the mainframe 360 and 370 models. It soon became obvious that exchange of information among computers and peripheral devices required standardization. What went on *inside* the computer was up to the vendor. What went on *between* computers and/or peripheral devices was to be standardized. ASCII (American Standard Code for Information Interchange) was developed and adopted by most computer companies as the standard. As its very name states, it is for **Information Interchange.** A manufacturer can still do anything he wants inside the computer — to maintain compatability with other computers, ASCII should be used in communicating with the outside world. ASCII is by far the most popular code, especially now that the IBM PC uses it. All the home computer systems use some variation of ASCII, internally as well as externally.

The typical personal computer's microprocessor handles numbers in eight-bit bytes. Eight bits are enough to handle 256 different codes — more than adequate for most microcomputer applications.

We will now explore how the character sets are handled on each of the major computer systems covered by MICRO.

Apple

The older Apple II uses a subset of standard ASCII code, in that some of the standard codes are not used. In models before the Apple IIe, the lower-case alphabet was excluded. When a character is put on the screen, this system is expanded to include flashing and inverse characters. On the Apple IIe, the upper 128 characters are almost standard ASCII (except for the fact that they have bit Hex 80 on), and the lower 128 are flashing and inverse versions, 64 characters each.

Atari

The Atari uses a system called ATASCII. Many standard characters are not implemented, and a number of graphic and cursor-movement characters have been added. Lowercase characters have been included for word processing. There is a reverse mode for screen and Atari printer display.

Commodore

Commodore systems, including PET, CBM, VIC, and Commodore 64, have 128 different characters (plus the 128 reverse images) that can be displayed on the screen. There are more than 128 codes (not all are displayable characters) that can be generated from the keyboard and handled by BASIC. The extra codes include cursor movements, color controls (on the VIC and C-64), and screen formating controls (on business machines). There are two complete character sets available (one at a time). One includes an extensive array of graphic characters, and the other includes lower case, at the expense of 26 graphic characters. PETSCII is a translation of ASCII codes to CBM codes. In lower-case mode the lower case letters are where ASCII upper-case is normally located and the upper case has the bit $80 set.

TRS-80 Color Computer

The Color Computer uses a subset of ASCII. Many standard control codes have been left out. There are 16 different block-style graphic characters available in each of eight different colors for a total of 128. There is no lower-case character set, but there is a mode where unshifted characters come out reversed and shifted characters come out normal. These will appear correctly on a printer. With the exception of the backspace, the arrow keys don't function for cursor movement, unless specifically programmed.

Hardware Characteristics Of Screen Display
or
More Than You Ever Wanted to Know About Your CRT

The method that your microcomputer uses to display characters and other graphic images on the CRT monitor or TV set is a fairly complicated procedure. Since the screen must be rewritten at the rate of 60 times a second to prevent screen flicker, the whole process of converting from computer codes to screen display character pattern must be done often and quickly.

The smallest screen display point is called a pixel (short for picture element). These pixels can be individually seen on a high quality monitor. The number of horizontal and vertical pixels is called the screen resolution. See Figure 1 for a list of the resolutions of typical microcomputers.

Figure 1

Microcomputer	Resolution
Apple	280 x 192
Atari	320 x 192
Color Computer	256 x 192
Commodore 64	320 x 200
Vic-20	184 x 176

Most microcomputers use memory mapped video display. This means that the current screen information is stored in random access memory. The RAM memory locations used for storing the current screen information is called the screen memory. Each position on the screen has a corresponding location in memory which holds the internal code for the character to be displayed at that position. (High-resolution graphics usually use some sort of encoding scheme so that several pixels can be stored in one byte. This is done to prevent the screen display from using all available RAM.)

Screen memory can be in a location fixed by hardware, such as the Apple, or can be pointed to with a screen address vector, such as the CoCo, Atari and Commodore computers. All computers have a standard default area for screen memory storage. See Figure 2 for these default locations.

Figure 2

Microcomputer	Screen Location	(Default)
Apple	$0400 - 07FF	Variable
Atari (48K)	$9C40 - A03F	Variable
Color Computer	$0400 - 05FF	Variable
Commodore 64	$0400 - 07FF	Variable
VIC-20	$1E00 - 1FF9	Variable

The system input routine normally handles the keyboard input and the placement of codes (including the cursor) into the proper locations of screen memory. This is generally a software routine and can normally be modified or replaced by the user. The screen display routine is more hardware oriented and is much more difficult for the user to change. The Apple is the least flexible in this respect, while the Atari has the most screen display options available of the microcomputers being discussed.

The microcomputer has special hardware, the display processor, to handle the generation of characters from internal codes, the refreshing of the video display, and other display oriented activities. The system microprocessor is normally *not* heavily involved in the continual maintenance and refresh of the display. During each raster scan (60 each second), the screen memory must be scanned by the display processor. Each memory location, including those containing blanks or spaces, is read and the code is converted into a bit-pattern that will generate the character for printing on the screen. A sample bit-pattern for a Commodore "A" is shown in figure 3.

Figure 3

```
...**...
..*..*..
.*....*.
.******.
.*....*.
.*....*.
.*....*.
........
```

Note: The asterisks represent "1" bits and the periods represent "0" bits.

The character memory has the bit-patterns for all of the characters that the computer can print on the screen. These patterns are normally stored in ROM. In some microcomputers, such as the Commodore and the Atari, the character memory is pointed to by an address in RAM, which means that this pointer can be modified to point to a start address in RAM, where you can define your own character set. In the case of the Apple and CoCo, the character ROM is hardware addressed. This means that the characters cannot be changed except by changing the character ROM itself.

The display processor must look up the first row of each character's bit-pattern and then display it on the first scan line of the screen. Then the row number is incremented and the routine looks up the second row pattern of each character and places that on the second line of the screen. This continues for all eight rows of the first line of characters and then for all character lines (16 to 25 depending upon the computer).

The Apple requires additional translation, since the screen is not contiguous in memory, but divided into three-part "macro-lines". In the case of the Commodore 64 and VIC-20, an additional lookup is required into the color memory to determine which color the character is supposed to be.

This process must be repeated 60 times a second. All screen updating is done during the vertical blanking period when the beam is moving back to the top left of the CRT in preparation for the next scan. This prevents the jibberish that would appear on the screen if a character should be changed during the screen refresh process.

(Continued on next page)

Screen Editors

Screen formats

Microcomputer	Horizontal	Vertical
C-64	40	25
Apple II	40	24
Atari	40	24
TRS-80C	32	16
VIC-20	22	23

A screen editor allows you to draw line images, character graphics images and character screen formats directly on the screen interactively with immediate feedback and correctability for fast, efficient generation of screen displays. Compared to programming PRINT statements with the accompanying running, editing and rerunning to work everything out, a screen editor allows more time to be spent on the actual looks of the screen and less time about how to generate it.

Each program allows the 'cursor' to move around the screen to be able to place characters anywhere on the screen independently of other characters already in place. These characters range from normal alphabetic characters to individually programmed characters which can be any shape possible in the matrix format of the computer. The Apple and TRS80 do not allow charcter redefinition, while the Commodore and Atari machines do. The cursor keys work on the computers that have them, other control keys are used when they are absent.

After entering the program listing from the section appropriate to your computer, save it to disk or tape and then type 'RUN'. You will be asked sveral startup questions such as SAVE or LOAD?, (answer S until you have some pictures on file), type of save desired (BASIC, textfile or binary), and prompted for a picture filename. Then the screen will be cleared and you have a blank screen — your sketch pad. Each computer has several function keys to change color, change mode (line drawing, graphics characters or alpha'numeric) and save or print. A complete description of the command keys and how each program works are in the individual sections with the program listings.

These programs include several different SAVE/LOAD methods and screen printing techniques. Different save options included, depending on the computer, are text file save of data statements, PRINT statement save of strings, and binary file save of the screen. These saves can be to tape or disk. Various loads are operable depending upon the save used. The data statements can be READ and POKEd, the PRINT statements can be RUN, or the binary file can be LOADed. Each method has its advantages and disadvantages depending upon the machine and the intended use.

The binary file save uses the start and end addresses of memory to save and dumps the screen information to disk or tape. The VIC and C 64 also have to save the color memory information. A data statement save writes a program of data statements that can be run to restore the screen information. Each piece of data is the contents of one byte of screen data. Again, the VIC and C 64 have to save color memory. This is a much slower method of screen save and restore. A BASIC program save to store the screen information in strings is the fastest method of restoring the screen. Each of the bytes of screen memory is processed and added to a string, one string for each screen line. You then save the program that will print these strings.

One feature that was not included in the programs that might be nice for a kaleidoscope effect would be to have the program print any character typed into one quadrant on the other three quadrants so that creating symetrical pictures would be very fast and easy. For Apple owners with lower case display, a modification to look at the shift key for upper/lower case input could also be added.

Programmable Character Generator

Someone, somewhere, during the design of your microcomputer, decided what each and every character that your system could generate would look like. Likewise, someone designed the character set for your printer.

The character generation capabilities of the microcomputer and the printer have both evolved over a long period of time. The original microcomputers, including the MITS/Altair, the IMSAI, KIM-1 and many others, had no video display capability. They each required a printing terminal — usually a Teletype. This, as mentioned earlier, was similar to a typewriter, and had only the letters of the alphabet, numbers, punctuation, and a few functions such as carriage return and line feed. Later models of the Teletype added lower case capability, but were still quite limited. Then SouthWest Technical Products introduced the video terminal. Originally it was nothing more than a *Glass Teletype*. It was faster, quieter and cost less to buy and operate — but it only had the limited character set of the Teletype. Even things which are taken for granted today, reverse video for example, were options for which you paid more or had to build yourself! In the beginning there was not even the lower case alphabet. It was mostly a matter of economics. Way back then (ca 1977), memory was still very expensive. The 2K EPROMs were just being introduced and cost upwards of $75.00 — *if* you could get them! RAM was even more costly. So, since the microcomputers at that time did not support any *fancy* characters, why spend money on unused terminal capability.

The picture began to change when the microcomputer added its own video. The Apple II and PET were pioneers in this field. Apple chose to go with only 64 characters — a subset of ASCII plus the same characters reversed and blinking. You see, it only took a few inexpensive gates (certainly under a dollar) to add the reverse and blinking capabilities. It would have taken much more to add ROM, EPROM or RAM. Then Apple put their money into high resolution graphics and color. The PET people made a different set of compromises that resulted in upper/lower case alphabetics and many graphic characters, but no color.

The next generation of personal computers — the Atari, TI, CoCo, VIC and Commodore 64 — each made significant additions to the character graphics. The additions included larger character sets (ROM and RAM were becoming cheaper); extensive use of color; multiple display modes; programmable characters and more. The printer companies also began adding graphics capabilities. Epson's original printer had essentially no graphics, until they manufactured an optional add-on. Other printer manufacturers started adding character-oriented features: multiple character sets

(Continued on page 16)

that could be selected under program control; some user-definable characters; 256 user-definable characters; the capability to copy the standard ROM characters into RAM to enable the user to modify selected characters and so on.

The result is that you are no longer dependent on a fixed set of graphics. It now makes sense for you to consider designing and using your own character sets. With the right microcomputer and printer, you can have almost any set of character graphics you desire, changing them as required and display them on both your CRT and printer. All you need is some easy way to design and generate your own characters. You can use DATA statements in BASIC, but computers are designed to work for you — not the other way around.

If you have an Atari, VIC or C64, then you can program characters on your system. If you have an Apple or CoCo, it requires changing a ROM to change the character set. However, some of what we are discussing is applicable to printers as well. After all, you will probably buy another microcomputer someday — so learn what is currently available.

Characters are normally stored in ROM in an 8 X 8 bit matrix of 8 bytes per character. The Atari has 128 different characters plus their inverse images. The Commodore machines have 256 different characters plus the inverse images. You can redefine characters by placing the character table in RAM and changing the pointer locations to reflect the new area to be utilized. Then by copying the old set and changing a few of the characters or creating a completely new set, you can print any type of character that you want on the screen, as long as it has an 8 X 8 format.

The Programmable Character Generator is an interactive program which lets you define new characters right on the video display, see how they look in actual size, and then save/load/modify them at desired. The PCG will operate somewhat differently on the Atari, VIC and C64, but these general operating instructions provide the basic guidelines.

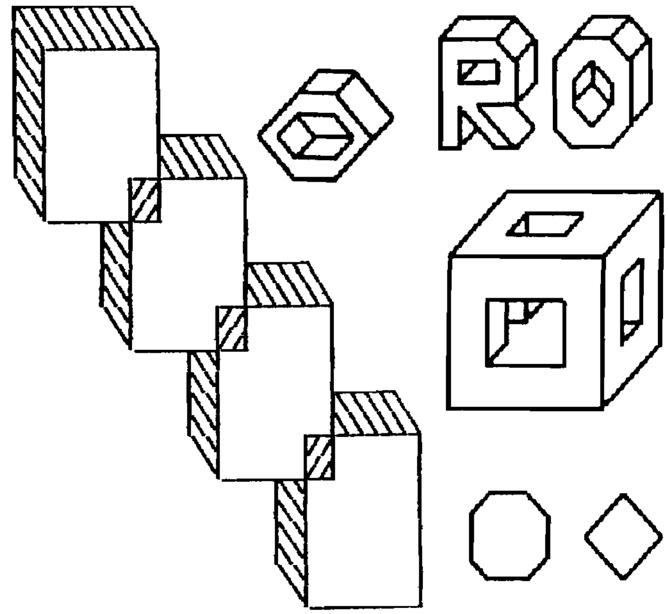
1. Type in the PCG program listed in your computer section and **save** it.
2. Type 'RUN'.
3. You will be presented with a display of the current characters, and an editing space in which to work on a new design.

There are two basic functions: Read and Change. Read permits you to examine an existing character in a larger format. Change allows you to modify or replace it with a new character. Each change modifies the current character and places it into the computers character RAM. The Editor allows you to move the cursor around in the limited character definition area and to set and clear the individual bits that will define your character.

The programmable character editor programs present an 8 X 8 block of dots on the screen — the filled in dots indicate that the bit is on, the hollow ones are off. The character is also presented in actual size so that you can see what the finished product will look like. By toggling the appropriate bits on and off, you can create a new character. By changing each character in the available set, you can create a whole new set of graphic characters or a different font style of letters and numbers.

These character sets can be SAVEd and LOADed so that you can use these character sets in your own programs, in the screen editor, etc.

Why should you need additional characters? You have probably lived this long without them — in fact, you have probably not made use of all the characters your computer gives you! Here are a few interesting characters that your computer does not have. The DRAWING program that is presented in a later section of this article requires certain characters for its significant use. The display printed below is made up of many characters which you do not have in your computer — yet!



The DRAW Program

The SCREEN Editor gave us the capability to put any character in any location of the display. This allowed us to make pretty pictures using the built-in character graphics of our computer. The Programmable Character Editor gave us the ability to re-define some or all of the standard characters into custom designed characters. With these two sets of tools, we were able to create some interesting pictures. However, it is still awkward to draw line-oriented pictures and diagrams since the cursor is always moving one position to the right, just as if we were typing text, even when we would rather not move the cursor in that direction. The DRAW Program solves this problem in two steps:

1. It defines a set of line graphics which can be used to draw charts, pictures and diagrams.
2. It provides special cursor controls that move the cursor in the direction of the line being drawn.

These two features working together in the DRAW Program, make drawing very easy.

This special drawing character set is made up of all possible combinations of four basic lines:

line	code
bottom horizontal	0001
left vertical	0010
left top to bottom right diagonal	0100
bottom left to top right diagonal	1000

(Continued on page 18)

VICTORY SOFTWARE

INTRODUCES

THE EARTH WARRIOR SERIES

METAMORPHOSIS $19.95

You stumbled into the nest of the Cyglorx and find yourself fighting off robot tanks guarding the Cyglorx eggs. You think you have everything under control and then the eggs start hatching. Commodore 64 *version features 4 different screens.*
Available for COMMODORE 64 and VIC-20. Played with JOYSTICK.

CREATOR'S REVENGE $19.95

The creator assembled a massive army of robots and insects to take revenge on the earth. Destroy insects, get treasures, and get the neutron bomb deactivator. Battle robots and destroy the neutron bomb before it annihalites your city. Miss and you must face the mutants. Features 4 different screens.
Available for COMMODORE 64. Played with JOYSTICK.

LABYRINTH OF THE CREATOR $19.95

Journey into the most complex and dangerous fortress ever build by the creator. You will encounter deadly robots, skulls, lakes, avalanches, false creators, and a creature who roams 256 rooms relentlessly pursuing you.
Available for COMMODORE 64. Played with JOYSTICK.

BOUNTY HUNTER $19.95

An adventure in the Old West. Journey back with us into the days of Jessie James and Billy the Kid where the only form of justice was a loaded revolver and a hangman's noose.

In this full-length text adventure, you play the role of Bounty Hunter, battling against ruthless outlaws, hostile Indians, wild animals and the elements of the wilderness with only your wits and your six gun. Average solving time: 20-30 hours. If you love adventures, this one is a real treat.
Available for COMMODORE 64 and VIC-20 (with 8K or 16K expander). Played with Keyboard.

CAVE KOOKS $14.95

Earth's surface is threatened by collapse from a strange group of creatures who bore out the earth's crust to make their dens. Your objective is to enter the creature's habitat and spin the invaders to death.
Available for COMMODORE 64. Played with JOYSTICK.

Programs for the VIC-20 and the COMMODORE 64.

All games available on TAPE or DISK.

All Arcade-Style games are packed full of MACHINE CODE.

KONGO KONG $19.95

Climb ladders, avoid the barrels the crazy ape is rolling at you, and rescue the damsel. Commodore 64 version features 4 different screens!

Available for COMMODORE 64 and VIC-20. Played with JOYSTICK or KEYBOARD.

GRAVE ROBBERS $19.95

Introducing the first GRAPHIC ADVENTURE ever available for the VIC-20 or COMMODORE 64! With realistic audio-visual effects, you explore an old deserted graveyard and actually see the perils that lie beyond.
Available for COMMODORE 64 and VIC-20. Played with KEYBOARD.

CHOMPER MAN $19.95

Don't let the bullies catch you as you gobble the goodies! This program has 8 screens and still fits in the standard memory.
Available for COMMODORE 64 and VIC-20. Played with JOYSTICK or KEYBOARD.

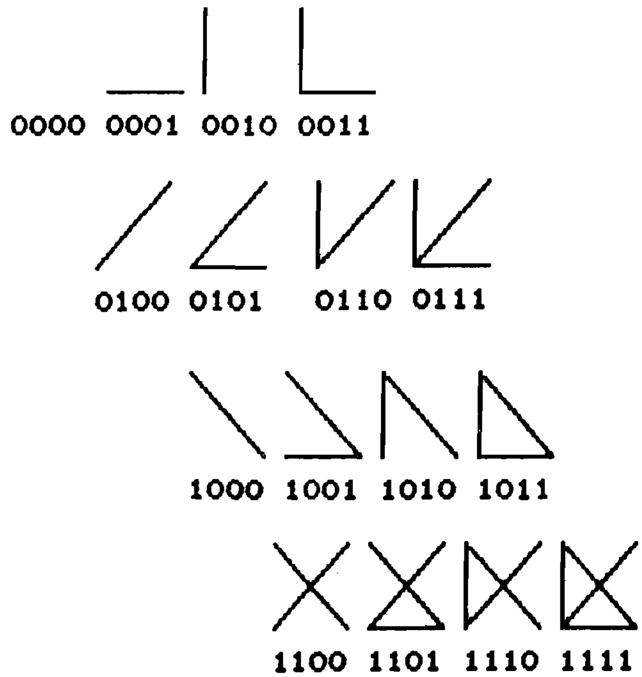
Illustrations: Elizabeth Hauck

Check your LOCAL DEALER or order directly.
ORDERING: We accept personal checks, money orders, VISA, and MASTERCHARGE. Charge orders please include number and expiration date. Overseas orders please use charge or have check payable through a U.S. bank. Add $1.50 postage and handling per order. PA residents please add 6% sales tax.

VISA — master charge

VICTORY SOFTWARE INC.
7 Valley Brook Road, Paoli, PA 19301
(215) 296-3787

Combinations of these lines generate sixteen characters:



This is a useful character set as it stands since it can be used to make line drawings such as these:

Two Dimensional Shapes:

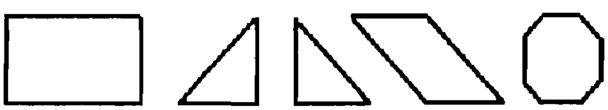
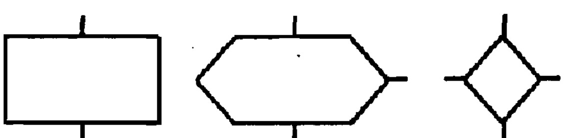
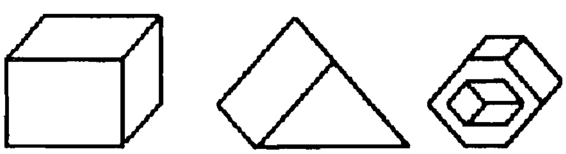


Chart Symbols:

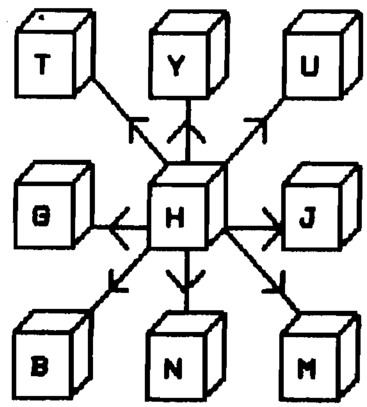


Three Dimensional Shapes:



The usefulness of this set of characters can be greatly enhanced by the DRAWING Program. With the normal SCREEN Editor, you have to position the cursor where you want it, decide which of the sixteen line combinations you want at that spot, figure out which key to press to generate that character, and then the cursor moves to the right when you press the correct key. There is a better way. The DRAWING Program redefines eight keys on the computer keyboard to represent each of the four independent line elements: North-South, East-West, Northeast-Southwest and Northwest-Southeast. In the diagram below, think of the letter H as being at the current cursor location. If you press the letter J, which is to the right of the letter H, then the cursor will move one position to the right and will draw a horizontal line. Press the letter G and the cursor will move to the left, and will draw a horizontal line segment. Likewise, the other letters surrounding the letter H will move the cursor in the same relative direction as that key is to the letter H, and draw the corresponding line segment.

Direction of Line Segments Relative to H:



As you press the key relative to the letter H, the cursor moves and draws a line segment in that direction. You can draw with lines in any direction — quickly and effortlessly!

What happens when there is already a line segment in the character space through which the cursor is moving? For example, what if there is already a horizontal line segment and the letter Y is pressed for a vertical up line? You then get the combination of the two individual segments. The characters are 'ORed' together.

A horizontal — plus a vertical |

generates a corner 

That is the reason that all sixteen possible combinations of the four line segments are defined. A corner is the combination of the horizontal and vertical, meeting in the corner.

While it is handy to be able to draw line segments to form a picture or diagram, it would also be nice to be able to erase individual segments to make alterations and corrections. The DRAW Program also provides this feature. The letter H is used to change the "color" of the pen from the current character color to the background color. Drawing with the background color effectively erases line segments. Note that the entire character position is not erased, only the line segment corresponding to the movement of the cursor. This makes it easy to erase a line and correct any mistakes. The normal cursor movement keys are left intact, so that you can move the cursor without drawing any lines.

Program Description

The DRAW Program is implemented as an addition to the existing SCREEN Editor. A line has been inserted in the SCREEN Editor after the main character input routine that goes to the DRAW support routines. These routines are logically divided into four parts.

The DISPATCH Routine tests for the DRAW or the EDIT mode key. If either is pressed, the correct mode is set. The mode is then tested, and if in Edit mode, an immediate return is made to the Editing routines. If it is in Draw mode, then the nine special Draw keys (the eight directions T Y U G J B N M and the draw/erase H) are tested. If it is not one of them, then a return in made to the Editor for normal operation. This permits the cursor controls and all keys other than the special nine to be used normally. All of the digits are available, for example. If one of the nine Draw keys is encountered, then calls are made to the DRAW LINE SEGMENT subroutines. Each of the eight Draw keys cause both movement of the cursor and writing or erasing a line segment.

The DRAW LINE SEGMENT routines consist of eight sets of calls to the MOVE and DISPLAY subroutines, one for each direction key. Each of the eight calls controls the generation of one of the eight line segment/direction combinations. For example, the line segment generated for the T key, that is the diagonal going Northwest from the current cursor location, is the series:

MOVE the cursor left one space
DISPLAY the Northwest diagonal character
MOVE the cursor up one space

The DRAW LINE SEGMENT subroutines provide this support service.

The MOVE subroutines handle the cursor movement. There are four basic directions: UP, DOWN, LEFT and RIGHT. Diagonal movements are made by moving both horizontally and vertically. The order of the movement and the displaying of the line segment are critical. For example, the letter T first moves the cursor left one position, draws the diagonal, then moves the cursor up one position. The letter U first prints the diagonal, then moves the cursor one position right and one position up. The MOVE subroutines control the cursor movement.

The DISPLAY subroutines control the displaying of the correct graphic character. If there is a character in the space to be written that is not one of the line segment characters or combinations, then it is erased. If the current location to be written in contains line segments, then the new segment is combined with the existing segments by a logical OR operation if the pen is down (writing) or is removed from the existing segments by a logical AND operation if the pen is up (erasing). A flag is set so that upon return to the normal Editing routines, a test will show that the character was a Draw command and that it has been serviced. The DISPLAY subroutines handle the actual output of the graphic character.

Operating Notes: for the Commodore 64 and VIC 20

1. The line segment is drawn with the current color of the cursor. If the cursor color is changed, then the line colors will be changed.

2. All of the cursor control movements, including HOME, CLR, INSERT and DELETE are valid during the DRAW mode. The only characters which are NOT available during DRAW mode are the letters that are used to control the Drawing: T Y U G H J B N M. These are available by going back temporarily to the Edit mode.

3. The cursor is changed in the DRAW mode to a small square. This represents the lower left corner of the character area and is the location where the horizontal, vertical and one diagonal meet.

4. The DRAW mode is called by the F7 key; the EDIT mode by the F5 key.

5. All of the normal graphic characters of the computer are available. The special line graphics have used the reverse forms of the characters:

@ A B C D E F G H I J K L M N O.

Typing Reverse A, for example, will generate the horizontal line segment.

Operating Notes for the Atari:

1. All of the cursor control movements, including INSERT and DELETE are valid during the DRAW mode. The only characters which are NOT available during DRAW mode are the letters that are used to control the Drawing: T Y U G H J B N M. These are available by going back temporarily to the Edit mode.

2. The DRAW mode is called by the double quote key (SHIFT 2); the EDIT mode is called by pressing the double quote key (SHIFT 2) when in DRAW mode.

3. All of the normal graphic characters of the computer are available. The special line graphics have used the characters:

! " # $ % & ' ( ) - + * , . / space

Typing Exclamation mark, for example, will generate the horizontal line segment.

The Screen Dumps

The programs also include a screen dump capability. They were tested using an Epson MX printer with Graftrax (both 80 and 100) except the Commodore programs use a Commodore printer (1525 or 4022). While the Commodore and Atari programs are slow running and take a while to print the whole screen, they do make an accurate representation of the screen image.

The Apple program dumps straight text, ignoring flashing and inverse characters. (It's hard to make a printer flash.) It PEEKs all of the screen memory locations, stores the codes in three integer arrays, and then dumps all of the character codes to the printer. This gives an idea of what was on the screen, but doesn't show inverse characters. With our new Epson FX80, we will be able to create an inverse character set so that they can be printed. We received the printer just before press time, so the inverse character program will have to be included at a later date.

The CoCo routine converts the 2 X 2 block matrix of character graphics to a corresponding 2 X 3 Epson TRS80 Model I equivalent and prints those along with the standard characters. Since the screen is contiguous in memory, it is easy to read one line at a time and dump it to the printer. There are two sets of data statements in the program to allow direct translation of block graphics into either 2 X 2 or 2 X 3 Model I block graphics.

The Commodore programs dump the strings to the printers in a manner similar to the CoCo, but includes inverse characters and all the Commodore graphics symbols that are built into the Commodore printers. The programs will not print any redefined characters, however.

The Atatri program (courtesy Paul Swanson) is the Mercedes of the print programs dumping exactly what is on the screen, redefined characters and all. If you always wanted to see all the character graphics in your program on your listing from an Epson, here is the program to do it. It switches the Epson to graphics mode and dumps the dot matrices from the character memory, either the standard ROM set, or a redefined set of your own making. It looks up the individual dot patterns and stores then in a large string, then dumping them to the Epson one pin line at a time, telling the printer exactly which print wires to fire in the graphics mode.

Using Templates in the Drawing Program

Once you start using the Line Drawing Program, you will discover that you often want to use the same basic shape again and again. Before long it may occur to you that there must be a better way than to re-type the all of these keystrokes. And there is. I will use the term 'template' to generically describe a number of techniques in which a set of characters can be automatically joined together to form a new entity which can be invoked with a single keystroke. The concept is not as difficult as it sounds. Suppose I draw a building made of bricks. This entails many keystrokes if I draw every single line for every single brick. Since the basic brick shape is repeated over and over, it would be easier and faster if an entire brick could be generated with a single stroke. The set of strokes to make a brick that is one unit tall and two units wide would be:

```
Y  J     J     N    G    G
up right right down left left
   >     >
   cursor right
```

This would generate the brick shape: 

If my program had a special key which invoked the line:

```
GOSUB UP:GOSUB RIGHT:GOSUB RIGHT:
   GOSUB DOWN
GOSUB LEFT:GOSUB LEFT: GOSUB
   CURSOR RIGHT
GOSUB CURSOR RIGHT: GOTO DONE
```

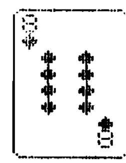
where UP, RIGHT, ... , are the line numbers of the line drawing routines to write a line and/or move the cursor in the specified direction, this would produce the shape:

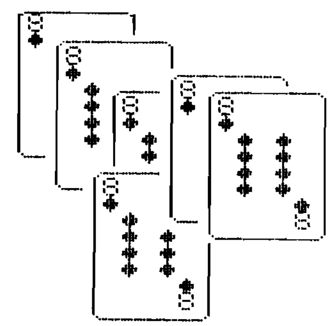
Since the brick drawing starts in the lower left corner and leaves the cursor in the lower right corner, I can draw a whole line of bricks simply by pressing my special key the desired number of times. If the key was Z, for example, pressing the letter Z six times would produce:

This simple concept of a pre-programmed shape can be expanded to include any other character graphics you desire. Another way to implement the same effect would be to write a string which had the correct characters in the correct sequence. The brick would be equivalent to the string:

```
BRICK$="YJJNGG"
```

A playing card on the Commodore would be:

Template 

Generated from Template 

There is essentially no limit to this capability. Any string you can store can be used to generate drawings, just as if they had been typed directly from the keyboard. These are all examples of fixed templates.

Variable Templates

Suppose that you wanted to use a lot of squares and rectangles in your drawing. You could define each possible size of square, rectangle or other shape that you plan to use, but that is tedious. Let the computer do the dull stuff! Examine the following lines:

```
IF A$="L" THEN LENGTH
IF A$="W" THEN WIDTH
IF A$="S" THEN SQUARE
IF A$="R" THEN RECTANGLE

LENGTH   GET A$:IF A$="" THEN
  LENGTH :REM GET DIGIT FOR LENGTH
LN=VAL(A$):IF LN<1 AND LN>9 THEN
   LENGTH
RETURN


WIDTH  GET A$:IF A$="" THEN WIDTH
WD=VAL(A$):IF WD<1 AND LN>9 THEN
   WIDTH
RETURN

SQUARE FOR I=1 TO LN: GOSUB UP:
   NEXT
FOR I=1 TO LN: GOSUB RIGHT: NEXT
FOR I=1 TO LN: GOSUB DOWN: NEXT
FOR I=1 TO LN: GOSUB LEFT: NEXT
RETURN
```

(Continued on page 22)

```
RECTANGLE   FOR I=1 TO LN:
        GOSUB UP: NEXT
  FOR I=1 TO WD: GOSUB RIGHT: NEXT
  FOR I=1 TO LN: GOSUB DOWN: NEXT
  FOR I=1 TO WD: GOSUB LEFT: NEXT
  RETURN
```

LENGTH and WIDTH simply allow you to set the length of a side or the width of a side with a value between 1 and 9. SQUARE draws a square with the length you previously set; RECTANGLE draws a rectangle with the height (length) and width that you set. You can now quickly and efficiently draw a square or rectangle anywhere on the display that you set your cursor. It sure beats doing it line character by line character. You can see how the basic concept can be extended to other shapes which have variable sizes within them.

User Generated Templates

As you play with the above templates and the line drawing program, you will undoubtedly reach a point, as I did, in which you say to yourself: "I wish new templates could be defined on the display as they were required." And that does make sense. Here is the code to do it:

Add to DRAW LINE Processing Routine:

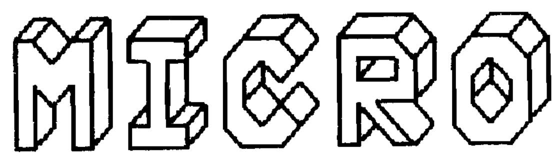
```
IF A$="!" THEN DF=0: GOTO PDONE :
  REM SAVE KEYS DONE
IF DF=1 THEN SS$=SS$+A$ : REM ADD
  CHARACTER TO STRING SAVE
IF A$="Y" THEN ... : REM NORMAL
  KEY PROCESSING

IF A$="C" THEN CAN-IT
IF A$="X" THEN REPLAY-IT

CAN-IT DF=1: REM SET SAVE
        KEYSTROKES MODE
SS$="" : REM CLEAR SAVE STRING
RETURN
REPLAY-IT   DF=2 : REM SET
        REPLAY MODE
FOR J=1 TO LEN(SS$) : REM DO ALL
        CHARACTERS
A$=MID$(SS$,J,1) : REM FETCH NEXT
        CHARACTER
GOSUB PROCESS : REM PROCESS
        CHARACTER
IF A$="" THEN RPDONE : REM DONE
        WHEN A$=""
PRINT A$; : REM NOT PROCESSED
        IF A$ <>""
RPDONE      NEXT : REM CONTINUE
        UNTIL END OF STRING SS$
DF=0 : RETURN : REM RESET FLAG TO
        NORMAL PROCESSING
```

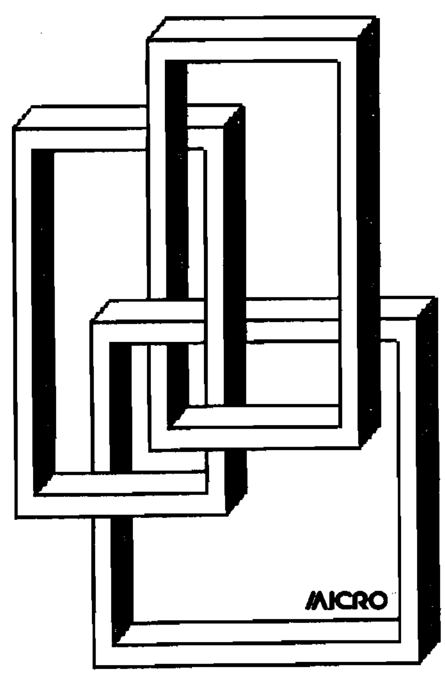
Now all you have to do to define one special template is to press the letter C to start the saving of every character typed. Type the drawing commands required to create your template. If you make mistakes, that is okay. Just correct them and keep on going. When you are done defining your special template, type the character ! to stop saving characters. Now you can position your cursor anywhere on the display, and press X and - there it is! Before your very eyes you will see your template drawn. And, you can repeat it as often as you like on the screen. The template does not have to consist of simple lines and cursor movements. You can type letters; other special graphics; the brick, square, rectangle fixed/variable templates; and so forth.

The concept can be expanded to permit multiple templates, user generated templates which call other user generated templates, and so on. A useful set that I created for my own computer has macro letters and number such as:



To create text using these letters, I simply type my special "start replay" key followed by the letter I need. And, the program uses the saved template to draw the letter — in three dimensions with shading!

There is an almost limitless number and variety of uses for the user defined templates. The current programs can be expanded to include saving and loading templates on tape or disk.



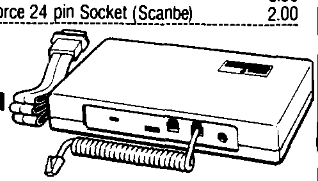
Hewlett Packard

Write or call for prices.

SPECIALS on INTREGATED CIRCUITS

6502	7.45	10/ 6.95	50/ 6.55	100/ 6.15
6502A/6512A	8.40	10/ 7.95	50/ 7.35	100/ 6.90
6520 PIA	5.15	10/ 4.90	50/ 4.45	100/ 4.15
6522 VIA	6.45	10/ 6.10	50/ 5.75	100/ 5.45
6532	7.90	10/ 7.40	50/ 7.00	100/ 6.60
2114-L200		2.45	25/ 2.30	100/ 2.15
2716 EPROM		4.45	5/ 4.20	10/ 3.95
2532 EPROM		5.90	5/ 5.75	10/ 5.45
6116 2KX8 CMOS RAM		6.45	5/ 6.20	10/ 5.95
4116 RAM				8 for 14
4164 RAM				6.50
Zero Insertion Force 24 pin Socket (Scanbe)				2.00

Anchor Automation Signalman Modems

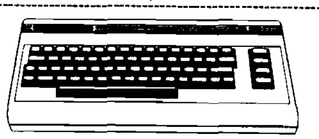


FREE SOURCE MEMBERSHIP WITH SIGNALMAN

All Signalman Modems are Direct Connect, and include cables to connect to your computer and to the telephone. Signalman Modems provide the best price-performance values.

Dealer and OEM inquiries invited

Volksmodem	(79)	59
RS232 or Atari Cable for Volksmodem		9
Mark I RS232	(99)	59
Mark VI for IBM Personal Computer	(279)	179
Mark VII Auto Dial/Auto Answer	(159)	89
Mark XII Smart Model 1200/300	(399)	295
DC HAYES Smartmodem		219
DC Hayes Smartmodem 1200/300		529



PROM QUEEN for VIC	170
Apple Emulator for Commodore 64	Call
STAT Statistics Package for C64	95
Solid Oak 2 Level Stand for C64 or VIC	29
C64/VIC Switch (networking)	125
BACKUP V1.0 tape copier for C64 or VIC	20
CARDBOARD/6 Motherboard - VIC	64
CARDBOARD/5 Motherboard - C64	56
CARD PRINT G Printer Int. with Graphics	72
CARD PRINT B Printer Interface—C64/VIC	40
CARDBOARD/3s Motherboard - VIC	32
CARDCO C64/VIC Calculator Keypad	32
CARDRAM/16 RAM Expansion - VIC	50
Complete CARDCO Line in stock	
CIE and VIE IEEE Interfaces in stock	
MAE Assembler for C64 + CBM	89

APPLE—FRANKLIN ITEMS

KRAFT Apple Joystick	40
Kraft Apple Paddle Pair	30
SPINNAKER Software in stock	
Broderbund Software in stock	
16K RAM Card for Apple	59
Multiplan—Microsoft	185
Solid Oak 2 Level Stand for Apple	29
Serial Card for Apple	99
MCP RAM/80 colemo card for IIe (AP/TXT)	139
Z80 Softcard and CP/M (Microsoft)	235
RANA Elite I with Controller	389
Parallel Printer Interface/Cable	79
Microtek Interfaces in stock	
Apple Dumpling with 16K Buffer	160
Grappler + Interface	129
Kraft Products for Apple in stock	
DC Hayes Micromodem II	299
PFS: File	95
PFS: Report	95
Videx 80 Column Card	209
Hayden Software for Apple 20% OFF	
Apple Blue Book	19

commodore

See us for Personal, Business, and Educational requirements. Educational Discounts available.

PETSCAN I $245 base price

Allows you to connect up to 30 CBM/PET Computers to shared disk drives and printers. Completely transparent to the user. Perfect for schools or multiple word processing configurations. Base configuration supports 2 computers. Additional computer hookups $100 each.

COMPACK/STCP $115

Intelligent Terminal Package for PET, CBM, C64

Includes ACIA Hardware / STCP Software

VIDEO ENHANCER for Commodore 64 69
Realize video quality equal or better than composite monitor using standard color TV.

SCREENMAKER 80 Column Adapter for C64 145
Provides big screen capability for business applications.

GENESIS Computer Corp

ViCcontroller (for C64 as well) combine with BSR modules for home or business control	50
COMSENSE Remote Sensing Adapter for C64 or VIC	35
COM VOICE Synthesizer for C64 or VIC includes software for test to speech, pitch, etc.	139
COM CLOCK Real Time Clock with battery backup	45

VIC 20 Products and Software in stock

Thorn EMI Software		**UMI Software**	
ABACUS Software		**HES Software**	
16K RAM for VIC	64	Vanilla Pilot	27

VICTORY Software for VIC and C64

Street Sweepers (VIC)	12	Kongo Kong (VIC)	16
Night Rider (VIC)	11	Cosmic Debris (VIC)	12
Annihilator	16	Adventure Pack I	16
Adventure Pack II	16	Metamorphosis	11
Educational Pack I	11	Trek	12
Strategy Pack I	16	Grave Robbers	12

PAPER CLIP Word Processor	109
ORACLE Data Base from Batteries Included	125
Super BusCard from Batteries Included	179
Commodore 64 Programmers Reference Guide	16
MicroChess for C64—8 levels of play excellent graphics and color	19
SPINNAKER Software C64, Apple, IBM, Atari	
Compute!'s First Book of PET/CBM	11
C64 or VIC SWITCH	125
POWER ROM Utilities for PET/CBM	78
WordPro 3+/64 with Spellmaster	85
WordPro 4+ - 8032, disk, printer	295
SPELLMASTER spelling checker for WordPro	
VISICALC for PET, ATARI, or Apple	189
PET-TRAX PET to Epson Graphics Software	40
SM-KIT enhanced PET/CBM ROM Utilities	40
Programmers Toolkit - PET ROM Utilities	35
EASY CALC for C64	65
PET Spacemaker II ROM Switch	36
COPYWRITER Word Processor for C64	69
2 Meter PET to IEEE or IEEE to IEEE Cable	40
Dust Cover for PET, CBM, 4040, or 8050	8
CmC Interfaces (ADA1800, ADA1450, SADI in stock)	
Programming the PET/CBM (Compute!) — R. West	20
Compute! First Book of VIC	11
OMNICALC (HES)	79
HES MODEM with Software	65
HES Software and Hardware in stock	
UMI products in stock	

FlexFile for PET/CBM/ C64 $110

Database, Report Writer with calculations, Mailing Lists.

FORTH for PET/C64 full FIG model — Cargill/Riley	$50
Metacompiler for FORTH for independent object code	30
KMMM PASCAL IV.1 for PET/C64	95
EARL for PET/CBM Disk-based ASSEMBLER	65
Super Graphics — BASIC Language Extensions Fast machine language graphics routines for PET/CBM	45
RAM/ROM for PET/CBM 4K $75	8K $90
Commodore Public Domain Software for C64	75

DISK SPECIALS

Scotch (3M) 5" ss/dd	10/ 2.20	50/ 2.00	100/ 1.95
Scotch (3M) 5" ds/dd	10/ 3.05	50/ 2.80	100/ 2.75
Scotch (3M) 8" ss/sd	10/ 2.30	50/ 2.10	100/ 2.06
Scotch (3M) 8" ss/dd	10/ 2.85	50/ 2.70	100/ 2.65

We stock VERBATIM DISKS

Write for Dealer and OEM prices.

Sentinal 5" ss/dd	10/ 1.90	50/ 1.85	100/ 1.80
Sentinal 5" ds/dd	10/ 2.55	50/ 2.50	100/ 2.45

We stock Dysan disks

Wabash 5" ss/sd	10/ 1.60	50/ 1.55	100/ 1.45
Wabash 5" ss/dd	10/ 1.90	50/ 1.85	100/ 1.75
Wabash 8" ss/sd	10/ 2.00	50/ 1.95	100/ 1.85

We stock MAXELL DISKS

Write for dealer and OEM prices.

Disk Storage Pages	10 for $5 Hub Rings 50 for $6
Disk Library Cases	8"—3.00 5"—2.25
Head Cleaning Kits	11

CASSETTE TAPES—AGFA PE-611 PREMIUM

C-10	10/ .61	50/ .58	100/ .50
C-30	10/ .85	50/ .82	100/ .70

DATASHIELD BACKUP POWER SOURCE 265
Battery back up Uninterruptible Power Supply with surge and noise filtering. The answer to your power problems.

Zenith ZVM-121 Green Phosphor Monitor	95
Zenith new color and monochrome monitors in stock	
MultiPlan—IBM or Apple	185
Quadboard for IBM available	
Peachtext 5000 Software Package	219
PFS Software for IBM and Apple in stock	
VOTRAX Personal Speech System	280
BMC 9191 Color Monitor	199
BMC 12A 12" Green Monitor	85
Dynax (Brother) DX-15 Daisy Wheel Printer	469
Itoh Prowriter Parallel Printer	379
Panasonic 1090 Printer with Correspondence Mode	279
USI CompuMOD 4 R F Modulator	39
Daisywriter 2000 with 48K buffer	1050
Many printers available (Gemini-Star, Brother, OKI, etc.)	
We Stock AMDEK Monitors	
Amdek DXY-100 Plotter	590
A P Products	15% OFF
Watanabe Intelligent Plotter 990 6-pen	1290
BROOKS 6 Outlet Surge Suppressor/Noise Filter	54
We stock Electrohome Monitors	
Synertek SYM-1 Microcomputer	189

ALL BOOK and SOFTWARE PRICES DISCOUNTED

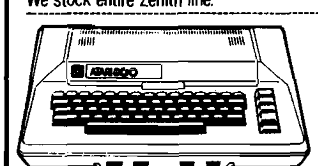
Panasonic 12" Monitor (20 MHz) with audio	129
Panasonic CT-160 Dual Mode Color Monitor	245

USI Video Monitors—Green or AMBER 20 MHz hi-res. Dealer and OEM inquiries invited

ZENITH data systems

HERO 1 Robot (factory assembled)	2145
Z29 Terminal (DEC and ADM compatible)	680
ZT-10 Intel. Terminal with Serial Port	340
Z100 16-bit/8-bit Systems in stock	CALL

We stock entire Zenith line.



ATARI SPECIALS

WE STOCK ENTIRE LINE—write for prices. SPINNAKER and Broderbund Software in Stock

215-822-7727
252 Bethlehem Pike
Colmar, PA 18915

A B Computers

WRITE FOR CATALOG. Add $1.50 per order for United Parcel. We pay balance of UPS surface shipping charges on all prepaid orders (add extra for mail APO/FPO, air). Prices include cash discount. Regular prices slightly higher. Prices subject to change.

INTRODUCTION TO COMMODORE GRAPHICS

by Loren Wright

Commodore Highlights

Commodore computers have one of the best systems of character graphics available. The PET has two character sets, one for graphics and the other for text. The VIC-20 and the Commodore 64 have the same character sets, with the addition of the British pound symbol (replacing the back slash). Characters on the VIC-20 may be in any of eight different colors, while the Commodore 64 has 16 to choose from. Also both machines have programmable character capabilities. In this issue I explore the character graphics capabilities of the Commodore machines in three different ways.

In my Commodore Compass column I present all-purpose routines for all three machines to plot with better-than-character resolution. The Commodore character set includes all the characters you need for single-pixel resolution line graphics or bar-graphs in either direction. In addition there is a full set of quarter-box characters that allow resolution to half a character in both directions at once. I have provided easy-to-use subroutines for each of these—just send the coordinates and color for each point, and the routine will do the rest!

The VIC/C64 Screen Editor program allows you the full screen to create character graphics — all available colors, reverse-field, insert, and delete. The programmable function keys are programmed to increment the screen and border colors. You can then SAVE your screen, complete with color information, for future use. You can also print the screen on a Commodore graphics printer.

Finally, there is a programmable character editor. With this program you can copy any number of the 256 characters into RAM, where you can change the pixel pattern. You can then SAVE the modified character set for future use. With a little extra programming on your part you can use such a modified character set in the Screen Editor.

In the following section, I discuss some of details of the character graphics capabilities of the various Commodore computers.

PET

The PET (and all its close relatives) have a rigid system of character graphics. The screen is located at $8000 (32768) and it consists of the 1000 or 2000 bytes from there on. It can't be moved. Any number that is POKEd into one of these locations instantly appears as a character on the screen. Since the PET/CBM is strictly black and white, there is no color memory.

There are two character sets programmed into a ROM — a graphics set that includes upper-case letters, and another set that includes both upper- and lower-case letters. The ROM contents are available only to the CRT controller. Even on the most recent releases, this ROM is socketed, so it may be replaced by an alternate character EPROM or by a board that switches between two or more character ROMs or EPROMs. Only one set of 256 characters is available at a time. On the PET models, the graphics set is the default; on the CBM, 8000 series, and SuperPET, the upper/lower case set is the default. POKE 59468,12 selects the graphics set and POKE 59468,14 selects the upper/lower case set. On the earliest PET models, the upper/lower case set worked backwards on the keyboard. Unshifted letters came out upper-case and shifted ones came out lower-case. The cure was, of course, a new character ROM!

VIC-20

The screen memory on the VIC-20 can be in a number of different locations, selectable by five bits in two different registers of the VIC chip (the VIC-20's CRT controller). In normal operation, though, there are only two locations used: $1E00 (7680) when the VIC has only 3K or no expansion and $1000 (1024) with 8K or more expansion. The screen memory consists of 512 consecutive bytes (506 of which actually appear on the screen).

Because the VIC-20 has color, there is an additional set of memory locations called *color memory*. Unlike all other RAM in the VIC, this area consists of 4-bit locations. Three of these bits are used to code for the eight different colors and the fourth bit is used to select multicolor mode. Like screen memory, there are 506 consecutive color memory locations. If you are working in the immediate mode or with PRINT statements that include color controls, this memory is managed automatically. If you are POKEing graphics to the screen, though, the corresponding color memory location must also be POKEd with the proper color. There are two possible addresses for this color memory: $9400 (37888) is used with 8K or more expansion, and $9600 (38400) is used with 3K or less. The VIC chip can only address 16K, and this accounts for the alternate locations.

The character ROM is visible to both the 6502 and to the VIC CRT controller. In addition, you can change the location where the VIC chip looks for its characters. From the 6502 perspective, the character ROM is at $8000 (32768). However, the pointer may be changed so that the VIC chip looks at RAM for the character pixel patterns. With 3K or less memory, the only useful locations for RAM characters are $1400, $1800, and $1C00. Because the

(Continued on page 26)

screen memory must start at $1E00 in this configuration, a complete 256-character set is only possible at the $1400 location. This leaves only 1K for your BASIC program in an unexpanded VIC and 4K with the 3K expander. With 8K or more expansion, the beginning-of-BASIC and other pointers may be changed before LOADing or entering any program. This leaves room for a full character set and the screen memory — all below $2000. All the expansion memory is available for a BASIC program. $1000 is also available as character RAM location, but extra manipulation is required.

In addition to the programmable character feature, there are three other character modes available. Double-height characters may be selected with a bit in a VIC chip register. In this mode, sixteen 8-pixel rows are put on the screen for each character. Character memory is also interpreted in 16-byte characters. Obviously, this mode is only useful in combination with RAM-defined characters. In fact, high-resolution graphics are accomplished by combining double-height characters with RAM-defined characters. Character memory (in RAM), in effect, becomes the high-resolution screen.

Another bit in the VIC chip controls reverse mode. When the bit is cleared, all characters on the screen appear reversed; when it is set, the characters appear normally.

Multicolor mode, mentioned above under color memory, is a special mode useful in combination with RAM-defined characters. In the normal mode, each bit in a character's definition determines whether a pixel on the screen is on or off. In multicolor mode, the pixels are twice the width of pixels in the normal mode and the bits in the character definition are considered in pairs. Together the two bits determine where to find the color of the double-width pixel on the screen. It is therefore possible to have four different colors in the space occupied by a character. See figure 1 for a diagram of a typical multicolor character. The four colors are 1. the screen color — 00, 2. the border color — 01, 3. the character color (determined by the lower three bits in the appropriate color memory location) — 10, and 4. the auxiliary color — 11. The auxiliary color is used only in multicolor mode, and is selected by four bits in a VIC chip register. Multicolor mode may be selected on or off for each screen memory location; it is controlled by the high-order bit in each color memory location. Eight different colors are available for the border and for each character, while sixteen different colors are available for the screen and auxiliary color. Changing the border, screen, or auxiliary color will result in an instant change in color of all the multicolor pixels specifying that register. In combination with the double-height character mode, multicolor characters defined in RAM may be used to produce a 4-color high-resolution screen.

Figure 1. Multicolor Character Mode a) Bits in character memory are considered in pairs. b) Each bit combination indicates a different source for the color. c) The final character displayed with double-width pixels.

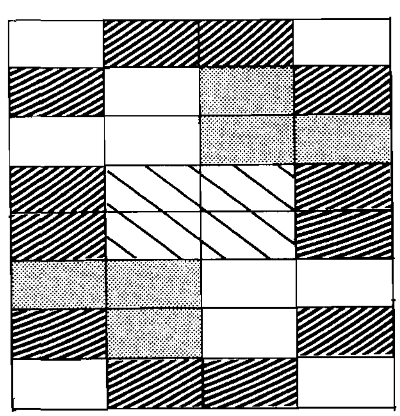
a)

0 0	0 1	0 1	0 0
0 1	0 0	1 0	0 1
0 0	0 0	1 0	1 0
0 1	1 1	1 1	0 1
0 1	1 1	1 1	0 1
1 0	1 0	0 0	0 0
0 1	1 0	0 0	0 1
0 0	0 1	0 1	0 0

b)

00	Screen Color
01	Border Color
10	Character Color
11	Auxiliary

c)



Commodore 64

The Commodore 64 is very similar to the VIC-20 in its character graphics capabilities. The actual implementation is more complicated and more difficult to program. Because the C-64 uses the 6510 processor with its tri-state address lines and built-in I/O port, some of the address space is shared among different functions. The I/O devices, such as the SID, VIC II, and CIAs, together with the color memory, share the same address space with the character generator ROM *and* with RAM. This means that when you want to copy characters from ROM to RAM, the I/O must be switched out and the character generator ROM switched in. (Normally the VIC II can see the character generator ROM, but the 6510 can't.) The VIC II also has only a 16K address space, so VIC II bank switching is necessary to achieve some results. Without switching banks, the most convenient place to store a programmable character set is at $3000. That leaves 10K for a BASIC program — adequate for most applications. The screen memory is normally addressed at $400, but it may be changed to any number of other locations, including $8000 for a PET simulation.

The overall character graphics capabilities are similar to the VIC-20's. Following are the differences:

1. There is no double-height mode.
2. There is no reverse mode.
3. There is an additional extended background mode. Only the first 64 characters are available, selected by the low-order six bits in screen memory. The two high-order bits select the register where the background color is stored. There are four background color registers in the VIC II chip — the screen color and three special registers. Extended background mode is selected for the whole screen by a single bit in a VIC II register.
4. Multicolor mode works similarly, but instead of the border color and auxiliary color, two of the special background registers of the VIC II are used. Multicolor mode is selectable only for the whole screen; on the VIC-20 each individual screen location may be selected.
5. High-resolution and multicolor bit-map modes are handled differently — not in the form of programmed characters.

Commodore Screen Editor

By Loren Wright

Program Operation

LOAD the screen-editor program and RUN it. First you will be asked whether you are using cassette or disk. Press the appropriate letter "C" or "D". Next, you will be asked whether you wish to LOAD or DRAW. Press the appropriate letter key, "L" or "D".

If you select "D", you will be asked for a name. This is the name under which your picture will later be saved, so be sure the name isn't one you have already used (unless you want to destroy it!). Then the screen will clear, with a white screen, red border, and black cursor in the upper left corner. You are now ready to draw.

If you select "L" you will be asked for a name. This is the name that will be searched for on the disk or cassette. Then you will be asked for a new name. If you don't have one, just hit RETURN. Use the new-name feature when you want to load a picture, modify it, and resave it as a new picture without destroying the first picture. The first name you supply will be used to locate the picture on the disk. The second name is the name used to store your picture when you are ready. If the two names are different, then you will end up with two different pictures on disk. Cassette operation is a little different, since location of a file is a long and tedious process. It is probably most efficient to use short tapes and store just one file per side. The way the program is written, the name is written to the cassette file, so the name you provide must match the name on the tape. The program may be easily modified to use no name for cassette files.

Creating a Picture

Generally, everything on the keyboard works — all the letters and graphics characters and the cursor movements. You can change colors by holding down the CTRL key and pressing the key with the proper color printed on the key-front. An additional eight colors are available on the Commodore 64 using the C= key with the 1, 2, 3, 4, 5, 6, 7, and 8 keys. The cursor will immediately change to this color. You can enter *reverse* mode by holding down CTRL and pressing "9" and exit it with CTRL and "0". (Unlike BASIC programming, you will stay in reverse mode until you exit it — even when you hit RETURN!)

The cursor keys move the cursor as indicated. In addition, if you try to move past the border, the cursor will appear on the opposite side of the screen. With a little practice, you will be able to move the cursor anywhere on the screen with a minimum of keystrokes. The only place you won't be able to move or type is the lower right corner.

Some Keys Have Special Functions or Peculiarities

INST Obtained with SHIFT and INST/DEL, this works similarly to how it works in BASIC. However, cursor movements will work normally, rather than appearing as reversed graphic characters. Most of the time characters will be bumped from the end of one line to the beginning of the next, but it will not *always* work that way, so don't rely on it. If you try to insert more than one character on the last line, the screen will scroll. The first character will be bumped into the otherwise inaccessible lower-right corner.

DEL This also works similarly to how it does in BASIC. Position the cursor to the right of the character(s) you want to delete and press DEL once for each delete operation. The same caution applies to bumping characters up from the next line: most of the time it will work but sometimes it won't.

f1 *Increment screen color.* Each time you press this key the screen color number will be incremented by 1. When it reaches 15, it will go back to 0. The first eight screen colors are the ones that appear on the 1,2,....8 keys. The others are listed in your *VIC-20 User's Guide* or *Commodore 64 User's Guide.*

f2 *Change file name to "TEMP".* This changes the current file name to "TEMP", protecting you in case you made a mistake in specifying the file name and didn't realize it until you had a creation ready to save. When you are done, you can rename the file to what you want.

f3 *Increment border color.* The border color is incremented by one each time this key is hit. On the VIC, only the first eight colors are available. On the C64, all sixteen colors are available.

f5 *LOAD file.* The LOAD sequence begins with a request for a file name. If the name isn't found, the screen will return cleared.

f6 *SAVE file.* The current screen will be SAVEd using the current file name. The name will be the one you typed in, or the name of the last file LOADed, or "TEMP" if you hit "f2".

f7 *PRINT Screen.* The cursor will disappear, and the screen will be sent to a properly interfaced Commodore graphic printer.

The Lower Right Corner

The lower right corner of the screen is inaccessible directly. However, there is a method to get a character there. Move the cursor to the next to last position, type the character you want to put in the corner, and press SHIFT and INST together *once only!* If you press INST again the whole screen will scroll up one line, most likely destroying a lot of hard work.

To remove a character from the lower right corner, move the cursor to the next to last position and press the DEL key.

How to Rename a File

If you have saved a file under the name "TEMP", then you will want to rename it to something that means a little more. Enter the following commands in the immediate mode (i.e., without your program running):

```
OPEN 1,8,15,"R0:newname = 0:TEMP"
CLOSE 1
```

Program Description

As you can see by the small number of changes required to run the editor on the VIC, nearly all of the program operation is independent of the machine. The differences are associated with screen size, color registers, and the locations of screen and color memory. Most of these differences are encountered in the form of variables set in lines 30 and 40. Instead of describing each variable individually at the beginning, I'll wait until these values are actually used.

Line 45 defines C$ as all of the color-control characters in the proper sequence. To speed things up, these individual characters are put into a string array C$( ).

In line 60 subroutine 4000 is called, which processes the initial prompts, file names, etc. Then the screen color is set to white and the border color is set to red. In line 70 the screen is cleared and the cursor color set to black. In line 90 we first encounter two heavily used variables: H for the horizontal position and V for the vertical position. These are both set to 0 to position the cursor in the upper left corner.

The main program loop begins at line 200. GOSUB 3000 extracts the current character and color information. Function T (for toggle, defined in line 20) takes a character code and converts it to *reversed* if it is *normal* and to *normal* if it is *reversed*. The result is returned in the variable X.

In line 210, the contents of memory location 646 are read into the variable C. This is the code for the current cursor color. GOSUB 1200 plots the reversed image of the current character in the current color. The GET T$ statement looks for a key from the keyboard. If none is there then T$ is filled with a null string. If this is the case, then the program loops back to the beginning of the line. This continues until a key is actually pressed, and the flow goes to the next line.

Lines 230-360 test for all the allowable control keys, and they are dispensed with appropriately. Before any action is taken, though, most of the keys cause the real character (as opposed to its reversed image) to be plotted at the current position. This is accomplished with a call to subroutine 1000. By following the flow, you should be able to tell what most of them do. The cursor-movement keys cause the appropriate modification in the value of H or V, followed by a jump to the limit-checking routine starting in line 600. If the program flow has gotten as far as line 370, then any control character will be printed. These are characters such as color controls that aren't dealt with individually. Nothing else needs to be done, since the cursor doesn't move for these characters, and the program can branch immediately back to the GET statement in line 210. All non-controls cause the program to drop through to line 500. At line 510 the character is printed and at line 520 the horizontal position is advanced by incrementing H.

Lines 600-670 make sure the cursor behaves properly when it gets to an edge. For instance, line 600 disallows an attempt to move to the lower right corner, and line 610 handles the left edge. Line 650 specifically deals with the insert and delete characters, cancelling the insert-mode flag with POKE 216,0. If the insert mode were left on, then a subsequent cursor move could appear as one of the strange graphic characters you are used to from BASIC listings.

Once the new position is adjusted to be within legal limits, the cursor is positioned according to the values of H and V (subroutine 2000) and the flow goes back to line 200, where the character at that position is reversed. Once again, the keyboard is checked for a key — and so on.

Plot character and color (1000-1040): This routine has two entry points. Most of the time it is entered at line 1000. In line 1010 X and C are set to the values for the actual character. In line 1200 (the alternate entry point) the actual position on the screen is calculated. Screen memory and color memory do not use the x-y coordinate system but instead are continuous memory locations starting at the upper left corner and ending at the bottom right. RW is the length of a row on the screen. PO is the calculated position, which is then added to the screen memory origin OG or the color memory origin CM.

Position cursor (2000-2010): The SYS call positions the cursor, using the value of the 65xx's X and Y registers. Not surprisingly, we have POKEd their shadows with the values of H and V.

Store current character and color (3000-3030): The position is calculated as in line 1010. Then, instead of POKEing, we PEEK to determine the character and color codes.

Initial prompts (4000-4040): First the CASSETTE/DISK choice is offered. If "C" or "D" is pressed, the device number DV is set appropriately. Then the LOAD/DRAW choice is offered. If "L" is selected, then the LOAD routine is entered. If "D" is selected, then the user inputs a name and a RETURN occurs.

SAVE routine (5000-5080): The actual SAVE operation is performed by the output routine (8000). Before the OUTPUT routine is called, the disk file name is modified with an appropriate suffix and prefix. Cassette file names are left unchanged, and the motor-stopping routine (9000) is called. The file is then opened using the proper device number, secondary address (same as the device), and name. DF is set to -1, signifying to the output routine that it was called from SAVE rather than DUMP SCREEN TO PRINTER. X% is calculated as a combination of the screen color and border color registers on the Commodore 64, or to the contents of 36879 on the VIC, which already represents both the screen and border color information. Using the same combination on the 64 allows for *some* compatibility between VIC and 64 pictures. X% is then PRINT#3ed to the file. Then the output routine is called to put out the entire screen with the exception of the lower-right character. Finally the PEEK/POKE values for the screen and color memory locations for the lower-right

are put out to the file before it is CLOSEd.

LOAD routine (6000-6110): The LOAD routine is much simpler and faster than the SAVE routine. First the screen/border color value is read and POKEd appropriately. Then the strings for each screen line are read and PRINTed to the screen. Finally, the lower-right corner values are read and POKEd appropriately before the file is CLOSEd.

Dump screen to printer (7000-7050): The work of this routine is primarily accomplished by the output routine. Line 7020 alters the line spacing on the CBM 4022 printer so that there is no space between lines. Consult the manual for your printer to adjust the line spacing appropriately. The flag DF is set to 0 so that the output routine will recognize that it was called by the printer routine. Before returning to the edit portion of the program, the file is CLOSEd.

Output routine (8000-8170): The function of this routine is to convert a screen line from the codes in screen (S%) and color memory (C%) to an equivalent string. Whenever the color changes, the appropriate color-control character must be inserted. The current color is maintained in the variable CF, and this is checked in line 8070 against each C%. If there is a change, the appropriate color-control character is tacked onto the string A$, and CF is changed to reflect the new value.

Reversed characters are identified by a 1 in the bit 7 position. This is tested from BASIC by ANDing with 128. If the bit is on the result will be 128; if it is off, the result will be zero. Dividing by −128 yields −1 when the bit is on and 0 when it is off. RV is the result of this operation, and RF always contains the result from the previous character. When both of these flags are the same then no change needs to be made. However, when they are different, an appropriate RVS or OFF character must be tacked onto A$. RF is set to reflect the new condition. This is accomplished in lines 8080 and 8085.

Now that the reverse/off information has been extracted, the code is ANDed with 127 to remove bit 7. Lines 8100-8125 convert S% to the appropriate PET-ASCII code. The CHR$( ) function is used to convert this code to a character and tack it onto A$ in line 8130.

Line 8145 tests if the routine was called by the printer or save routine. If it was called by the save routine and this is the last line, then the last character is removed with the LEFT$( ) function. In line 8150 A$ is put out to the file, which may be either the printer, the disk drive, or cassette.

Turn off cassette motor (9000-9040): Normally when cassette files are SAVEd the "PRESS PLAY &

(Continued on page 32)

Listing 1: **Screen Editor for Commodore 64. Marked lines are different for VIC-20 (see listing 2).**

```
 10 CR$=CHR$(13):DL$=CHR$(20):IN$=CHR$(148)
 20 DEFFNT(X)=(128+X)AND255
◆30 OG=1024:CM=55296
◆40 RW=40:MX=39:MY=24:SB=53280:SA=53281
◆45 C$="■■■■■■■■■■■■■■■■":L=LEN(C$)-1:DIMC$(L):FORI=0TOL:C$(I)
    =MID$(C$,I+1,1):NEXT

 50 X=160:CH=32
◆60 POKESB,2:POKESA,1
 70 GOSUB4000
 80 PRINT"■";
 90 PRINT"■":H=0:V=0
 200 GOSUB3000:X=FNT(CH)
 210 C=PEEK(646):GOSUB1200
 220 GETT$:IFT$=""THEN220
 230 IFT$="■"THENGOSUB1000:GOTO6000
 240 IFT$="■"THENGOSUB1000:GOTO5000
 245 IFT$="■"THENGOSUB1000:GOTO7000
 250 IFT$=DL$THENGOSUB1000:H=H-1:GOTO600
 260 IFT$=IN$THENGOSUB1000:GOTO600
 270 IFT$="■"THENGOSUB1000:H=H+1:GOTO600
 280 IFT$="■"THENGOSUB1000:H=H-1:GOTO600
 290 IFT$="■"THENGOSUB1000:V=V+1:GOTO600
 300 IFT$="■"THENGOSUB1000:V=V-1:GOTO600
 310 IFT$=CR$THENGOSUB1000:V=V+1:H=0:GOTO600
 320 IFT$="■"THENGOSUB1000:GOTO90
 330 IFT$="■"THEN80
◆340 IFT$="■"THENPOKESA,(PEEK(SA)+1)AND15
◆350 IFT$="■"THENPOKESB,(PEEK(SB)+1)AND15
 360 IFT$="■"THENNA$="TEMP"
 370 T=ASC(T$):IFT<32OR(T>127ANDT<160)THENPRINTT$;:GOTO210
 500 GOSUB2000
 510 PRINTT$;
 520 H=H+1
 600 IFH=MXANDV=MYTHENH=MX-1:GOTO660
 610 IFH<0THENH=MX:V=V-1
 620 IFH>MXTHENH=0:V=V+1
 630 IFV<0THENV=MY:GOTO600
 640 IFV>MYTHENV=0
 650 IFT$=DL$ORT$=IN$THENPRINTT$;:POKE216,0
 660 GOSUB2000
 670 GOTO200
 1000 REM PLOT CHARACTER AND COLOR
 1010 X=CH:C=CL
 1200 PO=V*RW+H
 1210 POKEOG+PO,X
 1220 POKECM+PO,C
 1230 RETURN
 2000 REM POSITION CURSOR
 2010 POKE781,V:POKE782,H:POKE783,0:SYS65520:RETURN
 3000 REM STORE CURRENT CHAR AND COLOR
 3010 PO=V*RW+H
 3020 CH=PEEK(OG+PO):CL=PEEK(CM+PO)
 3030 RETURN
 4000 PRINT"■■C■ASSETTE OR ■D■ISK"
 4010 GETT$:IFT$=""THEN4010
 4020 IFT$="C"THENDV=1:GOTO4100
 4030 IFT$="D"THENDV=8:GOTO4100
 4040 GOTO4010
 4100 PRINT"■L■OAD OR ■D■RAW"
 4110 GETT$:IFT$=""THEN4110
 4120 IFT$="L"THEN6000
 4130 IFT$="D"THENINPUT"■■NAME";NA$:RETURN
 4140 GOTO4110
 5000 REM DUMP TO CASSETTE OR DISK
 5005 PR$="@0:":EX$=",S,W"
 5010 IFDV=1THENGOSUB9000:PR$="":EX$=""
 5020 OPEN3,(DV),(DV),(PR$+NA$+EX$)
 5030 DF=-1
 5040 X%=((PEEK(SA)AND15)*16)OR(PEEK(SB)AND15):PRINT#3,X%
 5050 GOSUB8000
 5060 C%=PEEK(CM+PO)AND15:S%=PEEK(OG+PO):PRINT#3,C%:PRINT#3,S%
 5070 CLOSE3
 5080 GOTO90
 6000 REM LOAD SCREEN FROM TAPE OR DISK
 6005 EX$="":IFDV=8THENEX$=",S,R"
 6010 INPUT"■■NAME";NA$
 6015 INPUT"■NEW NAME■■■■■■";NB$:IFNB$="■"THENNB$=NA$
 6020 OPEN3,(DV),(DV-1),(NA$+EX$)
 6030 PRINT"■";
◆6040 INPUT#3,X%:POKESA,(X%AND240)/16:POKESB,X%AND15
 6050 FORY=0TOMY
 6060 INPUT#3,A$
 6070 PRINTA$;
 6080 NEXTY
 6085 PRINT"■";
 6090 INPUT#3,C%,S%:PO=MY*RW+MX:POKEOG+PO,S%:POKECM+PO,C%
 6100 CLOSE3
 6105 NA$=NB$
 6110 GOTO90
 7000 REM DUMP SCREEN TO PRINTER
 7010 OPEN3,4
 7020 OPEN6,4,6:PRINT#6,CHR$(18):CLOSE6
 7030 DF=0:GOSUB8000
 7040 CLOSE3
 7050 GOTO90
 8000 REM OUTPUT ROUTINE
 8030 FORY=0TOMY
 8040 A$="":RF=0:CF=-1
 8050 FORX=0TOMX
 8060 PO=RW*Y+X:C%=PEEK(CM+PO)AND15:S%=PEEK(OG+PO)
 8070 IFC%<>CFTHENA$=A$+C$(C%):CF=C%
 8080 RV=-(S%AND128)/128:IFRVAND(NOT(RF))THENA$=A$+"■"
      :RF=-1:GOTO8090
 8085 IF(NOT(RV))ANDRFTHENA$=A$+"■":RF=0
 8090 S%=S%AND127
 8100 IFS%<32THENS%=S%+64:GOTO8130
 8110 IFS%>63ANDS%<96THENS%=S%+128:GOTO8130
 8120 IFS%>95THENS%=S%+64
 8125 IFS%=32THENS%=160
 8130 A$=A$+CHR$(S%)
 8140 NEXTX
 8145 IFDFAND(Y=MY)THENA$=LEFT$(A$,LEN(A$)-1)
 8150 PRINT#3,A$
 8160 NEXTY
 8170 RETURN
 9000 REM TURN OFF CASSETTE MOTOR
 9005 REM  UNTIL KEYBD. KEY IS PRESSED
◆9010 POKE0,63:POKE1,PEEK(1)OR32
 9020 GETT$:IFT$=""THEN9020
◆9030 POKE0,47:POKE1,PEEK(1)AND223
 9040 RETURN
```

(Continued on next page)

RECORD..." message appears on the screen, and when the system detects a key pressed on the cassette machine, the SAVE operation begins. Since we don't want to mess up the screen with this message before we SAVE it, it must be suppressed. The solution is to turn off the cassette motor. It remains turned off until a keyboard key is pressed. This gives you a chance to press the appropriate cassette machine keys. When you hit a keyboard key the motor starts, the SAVE begins, and no message appears on the screen. The routine is different for the VIC and C-64.

***Listing 2:* Changes to listing 1 for VIC-20.**

```
30 OG=4*(PEEK(36866)AND128)+64*(PEEK(36869)AND112)
   :CM=37888+4*(PEEK(36866)AND128)
40 RW=22:MX=21:MY=22:SB=36879
45 C$="■▀◣▒▇":L=LEN(C$)-1:DIMC$(L):FORI=0TOL:C$(I)
   =MID$(C$,I+1,1):NEXT

60 POKESB,26

340 IFT$="▪"THENPOKESB,((PEEK(SB)AND240)+16)AND240OR
    (PEEK(SB)AND15)
350 IFT$="▀"THENPOKESB,(((PEEK(SB)AND7)+1)AND7)OR
    (PEEK(SB)AND248)

6040 INPUT#3,X%:POKE36879,X%

9010 POKE37148,PEEK(37148)AND241
9030 POKE37148,PEEK(37148)AND241OR14
```

MICRO

Commodore DRAW Implementation

by Bob Tripp

Make the changes and additions shown in listing 3. SAVE the completed program before you RUN it. For the VIC, you must make the screen editor VIC changes in addition to the ones shown in listing 4. The VIC version runs only with 8K or more expansion.

Operation is as described on page 18, except the following:

f7 Turn DRAW mode on and off.
f8 Begin storage of canned string.
f4 Used to generate dump to printer (instead of f7).

MICRO

***Listing 3:* Additions to C-64 Screen Editor program to implement DRAW.**

```
15 DF=0:MODE=0:PEN=1:XX=42
200 GOSUB3000:X=XX
225 GOSUB11400:IFA$=""THEN200
245 IFT$="▚"THENGOSUB1000:GOTO7000
645 IFFG=1THENFG=0:RETURN
665 IFDF=2THENRETURN
10000 REM ***** COPY ROM CHARACTERS **
10005 PRINT "GENERATING CHARACTERS"
10010 POKE 56334,REEK(56334) AND 254
10020 POKE 1,PEEK(1) AND 251
10030 FOR I=0 TO (256*8)-1:POKE I+12288,PEEK(I+53248):NEXT
10040 POKE 1,PEEK(1) OR 4
10050 POKE 56334,PEEK(56334) OR 1
10060 POKE 53272,(PEEK(53272)AND 240)+12
10070 PC=12288+(128*8)
10080 FOR I=PC TO PC+(16*8)-1:READ A: POKE I,A:NEXT
10090 RETURN
11200 DATA 0,0,0,0,0,0,0,0
11201 DATA 0,0,0,0,0,0,0,255
11202 DATA 128,128,128,128,128,128,128,128
11203 DATA 128,128,128,128,128,128,128,255
11204 DATA 128,64,32,16,8,4,2,1
11205 DATA 128,64,32,16,8,4,2,255
11206 DATA 128,192,160,144,136,132,130,129
11207 DATA 128,192,160,144,136,132,130,255
11208 DATA 1,2,4,8,16,32,64,128
11209 DATA 1,2,4,8,16,32,64,255
11210 DATA 129,130,132,136,144,160,192,128
11211 DATA 129,130,132,136,144,160,192,255
11212 DATA 129,66,36,24,24,36,66,129
11213 DATA 129,66,36,24,24,36,66,255
11214 DATA 129,194,164,152,152,164,194,129
11215 DATA 129,194,164,152,152,164,194,255
11400 REM  PROCESS DRAW COMMANDS
11404 A$=T$
11405 IF ASC(A$)=136 AND MODE=1 THEN MODE=0:XX=42:X=XX:GOSUB1200:RETURN
11410 IF ASC(A$)=136 AND MODE=0 THEN MODE=1:XX=123:X=XX:GOSUB1200:RETURN
11412 IF ASC(A$)=140 AND DF=0 THEN DF=1:SS$="":RETURN
11414 IF ASC(A$)=140 AND DF=1 THEN DF=0:RETURN
11416 IF DF=1 THEN SS$=SS$+A$
11420 IF MODE=0 THEN RETURN
11450 REM DECODE KEYS Y=N, U=NE, J=E, M=SE, N=S, B=SW, G=W AND T=NW
11451 IF A$="Y" THEN11481:REM NORTH
11452 IF A$="U" THEN11482:REM NE
11453 IF A$="J" THEN11483:PEM EAST
11454 IF A$="M" THEN11484:REM SE
11455 IF A$="N" THEN11485:REM SOUTH
11456 IF A$="B" THEN11486:REM SW
11457 IF A$="G" THEN11487:REM WEST
11458 IF A$="T" THEN11488:REM NW
11459 IF A$="H" THEN PEN = -PEN:GOTO 11660
11460 IF A$="S" TNEN 11700: REM SQUARE
11461 IF A$="Z" THEN 11710: REM BRICK
11462 IF A$="L" THEN 11720: REM LENGTH
11463 IF A$="W" THEN 11730: REM WIDTH
11464 IF A$="R" TNEN 11740: REM RECTANGLE
11465 IF A$="D" THEN 11750: REM DECISION
11466 IF A$="C" THEN 11760: REM CANNED STRING
11470 RETURN
11480 REM ROUTINES TO WRITE 8 DIRECTIONS
11481 GOSUB 11602:GOTO 11501
11482 GOSUB 11608:GOSUB 11502:GOTO 11501
11483 GOSUB 11601:GOTO 11502
11484 GOSUB 11503:GOSUB 11604:GOTO 11502
11485 GOSUB 11503:GOTO 11602
11486 GOSUB 11504:GOSUB 11503:GOTO 11608
11487 GOSUB 11504:GOTO 11601
11488 GOSUB 11504:GOSUB 11604:GOTO 11501
11500 REM MOVE CURSOR
11501 GOSUB1000:V=V-1:GOTO 11510
11502 GOSUB1000:H=H+1:GOTO 11510
11503 GOSUB1000:V=V+1:GOTO 11510
```

(Continued on next page)

Listing 3 *(continued)*

```
11504 GOSUB1000:H=H-1
11510 FG=1:GOSUB600: GOSUB2000
11520 GOSUB3000 :RETURN
11600 REM CALCULATE DRAW PATTERNS
11601 BP=1: GOTO 11610
11602 BP=2: GOTO 11610
11604 BP=4: GOTO 11610
11608 BP=8
11610 SX=128
11620 IF (CH>127 AND CH<144) THEN SX=CH
11630 IF PEN = 1 THEN CH=(SX OR BP)
11640 IF PEN = -1 THEN CH=(SX AND(-BP-1))
11650 CL=PEEK(646) : GOSUB1000
11660 A$="": RETURN
11700 REM S = SQUARE
11701 FOR I=1 TO LN:GOSUB 11481:NEXT
11702 FOR I=1 TO LN:GOSUB 11483:NEXT
11703 FOR I=1 TO LN:GOSUB 11485:NEXT
11704 FOR I=1 TO LN:GOSUB 11487:NEXT
11705 GOTO 11650
11710 REM Z = BRICK 1 X 2
11711 GOSUB 11481:GOSUB 11483:GOSUB 11483:GOSUB 11485:GOSUB 11487:GOSUB 11487
11712 GOSUB 11502:GOTO 11502
11720 REM GET LENGTH OF RECT.
11721 IF DF=2 THEN J=J+1:A$=MID$(SS$,J,1):GOTO 11723
11722 GETA$:IF A$="" THEN 11722
11723 LN=VAL(A$):IF LN<1 AND LN>9 THEN 11722
11724 IF DF=1 THEN SS$=SS$+A$
11725 GOTO 11650
11730 REM INPUT WIDTH FOR RECTANGLE
11731 IF DF=2 THEN J=J+1:A$=MID$(SS$,J,1):GOTO 11733
11732 GETA$:IF A$="" THEN 11732
11733 WD=VAL(A$):IF WD<1 AND WD>9 THEN 11732
11734 IF DF=1 THEN SS$=SS$+A$
11735 GOTO 11650
11740 REM RECTANGLE
11741 FOR I=1 TO LN:GOSUB 11481:NEXT
11742 FOR I=1 TO WD:GOSUB 11483:NEXT
11743 FOR I=1 TO LN:GOSUB 11485:NEXT
11744 FOR I=1 TO WD:GOSUB 11487:NEXT
11745 GOTO 11650
11750 REM DECISION BOX
11751 GOSUB 11488:GOSUB 11488:GOSUB 11482:GOSUB 11482
11752 FOR I=1 TO WD-2:GOSUB 11483:NEXT
11753 GOSUB 11484:GOSUB 11484:GOSUB 11486:GOSUB 11486
11754 FOR I=1 TO WD-2:GOSUB 11487:NEXT
11755 GOTO 11650
11760 REM CANNED STRING
11761 DF=2
11762 FOR J=1 TO LEN(SS$)
11763 A$=MID$(SS$,J,1):IFA$="C"THENJ=LEN(SS$):GOTO11769
11764 GOSUB 11450
11765 IFA$<>""THENT$=A$:GOSUB3000:GOSUB250
11769 NEXT
11770 DF=0:GOTO 11660
READY.
```

Listing 4: **VIC-20 changes to listing 1. Delete lines 10000-10090 and add the following. Remember to make VIC-20 changes shown in Screen Editor and to perform POKE 44,32; POKE 46,32; POKE 48,32; NEW before loading program.**

```
10000 REM ***** COPY ROM CHARACTERS **
10005 PRINT "GENERATING CHARACTERS"
10010 PRINT"♥":FORI=0TO255:POKE1024+I,I:POKE55296+I,0:NEXT
10020 POKE36869,PEEK(36869)AND240OR14
10040 FORI=0TO2047:POKE6144+I,PEEK(32768+I):NEXT
10050 FOR I=7168 TO 7168+(16*8)-1:READ A: POKE I,A:NEXT
10060 RETURN
READY.
```

MICRO

Programmable Character Editor for Commodore 64 and VIC-20

By Loren Wright

The *PC Editor* program copies all or part of a character set into RAM and then allows you to modify any character pixel by pixel. As you make the changes in the large display, the results will be reflected in the appearance of an actual-size character. When you are done, you may use a separate program to save the new character set to tape or disk in the format of your choice.

Because of the limited memory available on a 5K VIC, the VIC version copies only 192 characters (three-quarters of a full character set). You may modify the program to work with the full character set if you have more memory available in your VIC. The C64 version allows you to work with the full 256 characters.

Program Operation

Type in the program, SAVE it to tape or disk, and RUN it. With the Commodore 64 version, nothing will appear to happen for several seconds. With the VIC-20 version, the screen will fill from the top with random pixel data. As the character set is copied into RAM, the random dots will turn into recognizable characters. When this process is complete, the screen will clear and a large image of the "@" character will appear in the upper left corner. To the right you will see an actual-size "@" and below it a 0, indicating that we are currently displaying 0. Below this is a prompt display:

```
# TO ENTER CHAR #
F1 FOR NEXT
F2 FOR LAST
F7 TO EDIT
```

If you press the "#" key, you can enter the character code for any character you want to change. If you aren't familiar with these codes, they are *not* the same as the ones you get with the CHR$( ) function. Consult the PEEK/POKE Codes table in the back of your *User's Guide* for the proper codes. You may enter any number from 0 to 255.

The F1 and F2 keys get you a display of the next or previous character in the PEEK/POKE code sequence. When you finally have the character you want to change displayed, press the F7 key to enter the *edit* mode.

In the edit mode a new menu will appear, which is slightly different from the first menu:

```
+ ON        - OFF
F1 FOR NEXT
F2 FOR LAST
F7 TO SELECT
```

The editing cursor appears at the top left of the character display. Using the cursor control keys, you may move it around within the 8 × 8 space. A RETURN moves to the beginning of the next line, a HOME moves to the upper left corner, and a CLR (obtained with SHIFT and CLR/HOME) clears the current character. To turn on a pixel, press the "+" key; to turn off a pixel, press the "−" key or the SPACE bar. All of these keys will repeat if held down for more than a second. F1 and F2 will get you the next or previous character, but to select a character by number, you must return to the select menu. The F7 key switches to the select menu.

That's all there is to it. When you have finished modifying all the characters you want to change, you should type NEW and load in your data-saving program. You may save the memory space directly using Terry Peterson's BSAVE program (MICRO 64:96) or you may use the DATA-statement generator presented in this issue.

Special Notes for VIC – 20 with 8K (or more) Expansion

To take the best advantage of your extra memory, you should move the Start-of-BASIC pointer (and two others) to point just *above* where all the RAM characters are defined. You should delete line 5020 from the program. Also, you are now able to work with a full 256-character set:

```
5030 FOR I=0 TO 2047: POKE
NW+I,PEEK(32768+I): NEXT
```

Before you load the program enter the following line:

```
POKE 44,32: POKE 46,32:
POKE 48,32: NEW
```

Press RETURN, LOAD the program, and RUN it.

Cautions and Pitfalls

If you try to modify the normal letter and number characters, the listing and display will very quickly become unintelligible. It is better to modify characters that aren't used often such as the "@", British pound, and graphic characters. If you must change the solid-ball character used to create the large pixel display, you might want to change the display character to a "+" or another more common character. Changing the space character may also have disastrous results. Keep in mind that, although you may change the appearance of a character on the screen, you can't change how it works in the computer. Character #1 is always an "A" to the computer, even if you modify it to look like an "E".

If you press the RUN/STOP and RESTORE keys, you will change the computer so that it looks back at the character ROM for its characters instead of the RAM where you are making changes. Before you RUN the program, type GOSUB 5070 for the C64 and POKE 36869,254 for the VIC. The pointers that protect your character set from BASIC destruction remain intact even when you press the RUN/STOP and RESTORE keys together.

Another interesting, though harmless, effect of redefining characters is that the cursor will appear to behave peculiarly when positioned over a redefined character. It will flash between the new character appearance and a reversed image of the old character. This is because you haven't redefined the reversed character, which has a code 128 greater. When you are dealing with VIC characters with codes greater than 63, the cursor flash will make no sense at all. That is because the last 64 characters were not copied into RAM. On 5K VICs some characters on the screen may change with the cursor flash. That is because the character definitions for those characters are actually determined by the contents of the beginning of screen memory.

Program Description

Line 10 goes directly to the character-copying routine. When this is completed, the CLR wipes out all the variables defined so far. More important is that this forces the computer to readjust its pointers to protect the RAM characters from BASIC. This could not be done within the subroutine because the CLR also causes BASIC to lose track of all its FOR...NEXTs and GOSUB...RETURNs. The remaining lines (20-60) define the beginning of screen memory (OG), the beginning of color memory (CM), the beginning of RAM characters (NW), and the carriage-return character (CR$). Also, the screen is changed to a white background with a red border and the two functions we need in order to change a single bit are defined. FNA turns off a bit by ANDing the byte with a number consisting of all bits except the one that's changing. FNO turns a bit off by ORing the byte with a number consisting only of that bit. The argument of the function is the address of the byte, but the bit is determined by the horizontal position of the cursor. The cursor may take a horizontal position of 0-7 from the left size. Subtracting this number from 7 results in the correct bit number. Finally, the screen is cleared and the cursor color is set to black.

Lines 5 and 6 in the VIC version are required because the location of screen memory and color memory vary with the VIC's memory configuration. As mentioned above, these variable assignments are lost when the CLR command in line 10 is executed, and variables OG and CM must be recalculated.

The main program (lines 100-200) actually includes the select mode, since it is relatively simple. In line 100, CH (the character number) is set to 0 and the display routine (1000) is called. This displays the "@" sign. The call to subroutine 4000 in line 105 prints the prompt lines below the character display. Line 110 is actually the beginning of the select mode. This line is looped on continuously until a character is pressed. The only four allowable characters are tested in the next four lines; any other character is dealt with by the GOTO 110 in line 160.

Line 120 tests for the "@" character, which goes to subroutine 3000 for the user to input a character number. Line 130 tests for the F7 key, which causes a call of subroutine 4500, where the edit mode prompts are displayed. Line 140 tests for the F2 key, which causes the character number to be decremented. ANDing the new CH with 255 keeps the character number in the 0-255 range. Line 150 tests for the F1 key, which increments CH, again keeping it in range. All keys that change the character number result in a branch to line 170, where the display routine is called before branching back to line 110 for a new character. Line 190 is reached when the F7 key is pressed. First the new prompts are displayed, then the edit mode is entered with a call to subroutine 2000.

The PC Display routine (1000-1120) prints the pixel pattern of a character in a large 8 × 8 format using the period and solid-ball characters. In line 1010 the byte corresponding to the beginning of the current character is calculated. In line 1020 the cursor is moved to the upper left corner. Then nested FOR... NEXT loops examine each bit, printing a solid ball if it is on and a period if it is off. The PRINT statement in line 1090 moves the cursor to the beginning of the next line before starting the display of a new character row.

Lines 2000-2550 are the edit-mode subroutine. The RETURN is at the end of line 2070 — executed when the F7 key is hit. Lines 2010 and 2020 position the cursor in the upper left corner. The variable H and V are used to keep track of the cursor position. PO is the current screen address of the cursor, calculated using the origin OG, the length of a screen row RW, and the position variables V and H. BY, as in the display routine, is the address of the current byte, calculated using the beginning of the RAM characters NW, the character number CH, and the vertical position V. The POKE statement in line 2050 places a reversed image at the current cursor position.

The main edit routine begins in line 2060 with a GET loop. Line 2070 checks for the F7 key, RETURNing to the select mode. Lines 2080-2110 test for the cursor-movement keys, modifying H and V appropriately, followed by a branch to the limit-checking routine at line 2500. The carriage return (line 2120) moves the cursor to the beginning of the next line and branches to the vertical portion of the limit-checking routine. The F1 and F2 keys are checked in lines 2130 and 2140 and cause a display of the new character to appear before a branch back to the home-cursor point at line 2010. A HOME key in line 2150 causes the current position to be turned back to normal before branching to line 2010. The CLR key causes the eight bytes of the current character to be blanked before performing the display and branching back to the home-cursor point. The "+" and "−" keys use the two defined functions to set or clear the appropriate bit. Then H is incremented before a branch to the limit-checking routine at line 2500. The limit-checking routine checks H and V and adjusts them accordingly. The current position is returned to normal in line 2540 before a branch back to line 2030, where the newly calculated position is reversed.

Subroutine 3000 handles user input of a character number. If the number input is too large or small, the AND 255 will help prevent a crash. Line 3020 blanks out the prompt and the user's input.

Subroutines 4000 and 4500 print the appropriate prompts for the select and edit modes. Notice the extra spaces in lines 4030 and 4500 that overwrite the previous display. These two displays have to overlay each other without clearing the screen.

Subroutine 5000 is different for the two machines. The details are explained in the programmable characters section of your *Programmer's Reference Guide*.

MICRO

***Listing 1:* PC Editor for Commodore 64**

```
10 GOSUB5000:CLR
20 POKE650,128
30 OG=1024:CM=55296:RW=40:NW=12288
40 CR$=CHR$(13):POKE53281,1:POKE53280,2
50 DEFFNA(X)=PEEK(X)AND(255-2↑(7-H)):DEFFNO(X)=PEEK(X)OR2↑(7-H)
60 PRINT"▚■";
100 CH=0:GOSUB1000
105 GOSUB4000
110 GETT$:IFT$=""THEN110
120 IFT$="#"THENGOSUB3000:GOTO170
130 IFT$="▌"THENGOSUB4500:GOTO190
140 IFT$="▜"THENCH=(CH-1)AND255:GOTO170
150 IFT$="▆"THENCH=(CH+1)AND255:GOTO170
160 GOTO110
170 GOSUB1000
180 GOTO110
190 GOSUB2000
200 GOTO105
1000 REM PC DISPLAY
1010 BY=NW+8*CH
1020 PRINT"▙";
1030 FORI=0TO7
1040 X=PEEK(BY+I)
1050 FORJ=7TO0STEP-1
1060 IF(2↑J)ANDXTHENPRINT"●";:GOTO1080
1070 PRINT".";
1080 NEXTJ
1090 PRINT
1100 NEXTI
1110 POKEOG+RW+13,CH:POKECM+RW+13,0:PRINT"▙"TAB(11)"▆▆▆▆"CH"▌▌  "
1120 RETURN
2000 REM EDIT
2010 H=0:V=0
2020 PRINT"▙";
2030 PO=OG+V*RW+H:X=PEEK(PO)
2040 BY=NW+CH*8+V
2050 POKEPO,XOR128
2060 GETT$:IFT$=""THEN2060
2070 IFT$="▌"THENPOKEPO,XAND127:RETURN
2080 IFT$="▌"THENH=H+1:GOTO2500
2090 IFT$="▌"THENH=H-1:GOTO2500
2100 IFT$="▙"THENV=V+1:GOTO2500
2110 IFT$="▜"THENV=V-1:GOTO2500
2120 IFT$=CR$THENH=0:V=V+1:GOTO2530
2130 IFT$="▆"THENCH=(CH+1)AND255:GOSUB1000:GOTO2010
2140 IFT$="▜"THENCH=(CH-1)AND255:GOSUB1000:GOTO2010
2150 IFT$="▙"THENPOKEPO,XAND127:GOTO2010
2160 IFT$="▚"THENFORI=0TO7:POKENW+8*CH+I,0:NEXT:GOSUB1000:GOTO2010
2170 IFT$="+"THENX=81:POKEBY,FNO(BY):H=H+1:GOTO2500
```

(Continued on next page)

Listing 1 *(continued)*

```
2180 IFT$="-"ORT$=" "THENX=46:POKEBY,FNH(BY):
     H=H+1:GOTO2500
2500 IFH<0THENH=7:V=V-1
2510 IFH>7THENH=0:V=V+1
2520 IFV<0THENV=7
2530 IFV>7THENV=0
2540 POKEPO,XAND127
2550 GOTO2030
3000 INPUT"[reverse-video control characters]":CH
3010 CH=CHAND255
3020 PRINT"[reverse-video control characters]                    "
3030 RETURN
4000 PRINT"[reverse-video control characters]# TO ENTER CHAR #"
4010 PRINT"[reverse-video control characters]F1 FOR NEXT"
4020 PRINT"[reverse-video control characters]F2 FOR LAST"
4030 PRINT"[reverse-video control characters]F7 TO EDIT   "
4040 RETURN
4500 PRINT"[reverse-video control characters]+ ON       - OFF    "
4510 PRINT"[reverse-video control characters]F1 FOR NEXT"
4520 PRINT"[reverse-video control characters]F2 FOR LAST"
4530 PRINT"[reverse-video control characters]F7 TO SELECT"
4540 RETURN
5000 PRINTCHR$(142)
5010 POKE52,48:POKE56,48
5020 POKE56334,PEEK(56334)AND254
5030 POKE1,PEEK(1)AND251
5040 FORI=0TO2047:POKEI+12288,PEEK(I+53248):NEXT
5050 POKE1,PEEK(1)OR4
5060 POKE56334,PEEK(56334)OR1
5070 POKE53272,PEEK(53272)AND240OR12
5080 RETURN
READY.
```

***Listing 2:* Patches to listing for VIC-20**

```
5 OG=4*(PEEK(36866)AND128)+64*(PEEK(36869)AND112)
  :NW=6144
6 CM=37888+4*(PEEK(36866)AND128)
10 GOSUB5000:CLR
20 POKE650,128
30 OG=4*(PEEK(36866)AND128)+64*(PEEK(36869)AND112)
   :NW=6144
35 CM=37888+4*(PEEK(36866)AND128)
40 CR$=CHR$(13):POKE36879,26:RW=22
50 DEFFNA(X)=PEEK(X)AND(255-2↑(7-H)):
   DEFFNO(X)=PEEK(X)OR2↑(7-H)
60 PRINT"[reverse-video control characters]";
5000 PRINT"[reverse-video control characters]":FORI=0TO255:POKEOG+I,I:POKECM+I,0:
     NEXT
5010 POKE36869,PEEK(36869)AND240OR14
5020 POKE52,24:POKE56,24
5030 FORI=0TO1535:POKENW+I,PEEK(32768+I):NEXT
5040 RETURN
```

***Listing 3:* DATAMAKER for use with PC Editor**

```
100 INPUT"NC,LN,MS":NC,LN,MS
500 PRINT"[reverse-video control characters]";:FORI=CHTOCH+7
510 IFI>=NCTHENI=I+7:STOP
520 PRINTMID$(STR$(LN+I*10),2);"DATA";
530 FORJ=0TO7
540 PRINTMID$(STR$(PEEK(MS+I*8+J)),2);CHR$(44);
550 NEXTJ:PRINTCHR$(20)
560 NEXTI:CH=CH+8
570 PRINT"NC="NC":LN="LN":CH="CH":MS="MS":GOTO500
580 POKE631,19:FORI=1TO9:POKE631+I,13:NEXT:
    POKE198,10
```

Commodore Buyer's Guide

Unless otherwise indicated, the opinions represented here are those of Contributing Editor Jim Strasma. Jim is editor of *The Midnight/PAPER*, which specializes in reviews of products for Commodore computers. Although not covered by our categories for this guide, I strongly recommend *The Midnight/PAPER* for anyone making software and hardware purchasing decisions. US subscriptions, mailed first class, are $20; Canadian is $25; other countries: surface — $30, air — $35. Mailing address: *The Midnight/PAPER*, c/o Jim Oldfield, 635 Maple, Mt. Zion, IL 62549.

We also received opinions on VIC 20 and Commodore 64 products from Dave Malmberg, and in some areas I have provided some input.

Loren Wright

VIC-20

Games:
Jim's choices: 1) Choplifter, Creative, $40, 2) Serpentine, Creative, $40, and 3) Shamus, HES, $35.
Dave's choices: 1) Exterminator, Nufekop, $24.95, 2) Omega Race, Commodore, $14.95, and 3) Lazor Zone, HES, $29.95.

Educational:
Jim: 1) Pipes, Creative, $40, 2) Instrument Flight Simulator, Academy, $40, 3) Touch Typing Tutor, Taylormade, $19.95.
Dave: 1) Turtle Graphics, HES, $39.95, 2) Vanilla Pilot, Tamarack, $29.95, and 3) Touch Typing Tutor.

(*Ed. note:* Dave is the author of Turtle Graphics, but I can also highly recommend the package. *LW*)

Business Software:
1) The Complete Personal Accountant (formerly The Color Accountant), Programmer's Institute, $74.95 cass., $79.95 disk, 2) TOTL Time Manager 2.1, TOTL, $30, 3) TOTL Business 3.0.

Word Processors:
Jim: Wordcraft 20, UMI, $100 and $200, 2) Write Now!, Cardco, $39.95, and 3) TOTL Text, TOTL, $35.
Dave: 1) HES Writer, HES, $39.95, and 2) TOTL Text.

Graphics Packages:
1) Super Expander, Commodore, $39.95, 2)Graphvics, Abacus, $24.95, 3) Jim: VIC Pics, Midwest, $29.95, Dave: Grafix Designer, Midwest, $14.95. *(Continued on page 42)*

MICRO™

Commodore Compass

by Loren Wright

Low-Resolution Graphics on Commodore Computers

Most computers can do the kind of graphics I demonstrated last month in my article "Plotting Data with Character Graphics (MICRO 65:28)." The computer needs only a memory-mapped screen and some means of POKEing characters into that memory. Many computers, including the Commodore 64 and to a limited extent the VIC-20, have a full high-resolution graphics system, where every pixel on the screen may be set or cleared. This capability has a price, both in memory consumption and in flexibility. It takes 8K of memory just to store the pixel information for the typical 1000-character screen! Many computers, including the Apple and Atari machines, have some system of low-resolution graphics — a system in between character and high-resolution graphics. Such a system gives you better resolution than with character graphics and usually more convenient commands. The memory consumption is not as high as with high-resolution and usually more flexibility is available. The Atari, in fact, has a number of these modes, offering different resolutions, different restrictions on colors and luminances, and different memory requirements. The Commodore computers have no such system, but the rich variety of character graphics available makes it easy to do a number of low-resolution jobs, using much less memory than even the lo-res modes of Apple and Atari. On the Commodore 64 and the VIC, you get even more colors than with either Atari or Apple!

The following sample low-resolution drivers will work on any Commodore system. In the first two lines of the program, the screen memory origin, maximum X, maximum Y, and number of characters in a row are all set according to your particular machine. In addition, the VIC-20 and Commodore 64 require the color memory origin and initial color value. With one further exception, the rest of the program is Commodore machine-independent. The exception is that on the PET and CBM machines, you must omit any POKEs to color memory. If you leave one in by accident, you may get strange results, including a crash of your machine that can be recovered only by turning the power off and on again! Those who read last month's article should be familiar with my conventions by now. In the first few lines (10-99) constants and functions are defined, arrays are dimensioned, screen and border colors are set, and the screen is cleared. Lines 900 and 999 have the effect of freezing the screen after you are through graphing your data. Beyond that, beginning at an even 1000, is the plot subroutine. At the beginning of each subroutine covered, in a REM statement I list the input variables and their ranges. The actual function you are plotting is usually contained in only three or four lines numbered between 100 and 899.

Easy Bar Graphs

The Commodore character set includes all the characters you need to draw bar graphs to a single-pixel resolution in either horizontal or vertical directions. Listing 1 is a program that draws 25 bars of random lengths and colors from the left edge of the screen. The subroutine, beginning in line 1000, requires a row number (from 0 to 24) and a length (adjusted to the range 0-319). Instead of the random sequence, place your own program starting at line 100.

You may wonder for a moment why you don't see all the characters used at the end of the bars on the keyboard. The first four characters *are* on the keyboard, but the last four characters are actually reversed images of other characters.

***Listing 1:* BARPLOT for Commodore 64**

```
10 MX=319: MY=24: RW=40: OG=1024: CM=55296: NC=16
30 PRINT"♥"
100 FOR Y=0 TO MY
110 X=RND(1)*MX
120 CC=RND(1)*NC
130 GOSUB 1000
140 NEXT Y
900 GET T$: IF T$="" THEN 900
999 STOP
1000 REM BARPLOT ROUTINE
1010 REM CC=COLOR: X=0 TO MX: Y=0 TO MY
1020 IFBFTHEN1100
1030 RESTORE: FORI=0TO8: READBP(I): NEXT: BF=-1
1100 IF X=0 THEN 1150
1110 X0=XAND7: X1=INT(X/8): B0=Y*RW
1120 IF X1=0 THEN II=0: GOTO 1140
1130 FOR II=0 TO X1-1: POKE OG+B0+II,160: POKE CM+B0+II,CC: NEXT II
1140 POKE OG+B0+II,BP(X0): POKE CM+B0+II,CC
1150 RETURN
9000 DATA 32,101,116,117,97,246,234,231,160
```

Line changes for VIC-20:

```
10 MX=175: MY=22: RW=22: NC=8: OG=4*(PEEK(36866)AND128)+64*
   (PEEK(36869)AND112)
20 CM=37888+4*(PEEK(36866)AND128)
```

Line changes for PET:

```
10 MX=319: MY=24: RW=40: OG=32768

1130 FOR II=0 TO X1-1: POKE OG+B0+II,160: NEXT II
1140 POKE OG+B0+II,BP(X0)
```

(Continued on next page)

Extra Vertical Resolution

Listing 2 is a program that draws a sine-squared plot using only the eight different horizontal line characters. If you have the programs from last month's article, compare this sine-squared plot to the one we generated there. Note the smoother appearance at the peak and at the tails. Of course, the disadvantage is apparent in the straighter stretches in between. (Because of legibility problems with Commodore 64 characters containing *vertical* lines, most characters containing *any* single lines have been changed to double-pixel width lines. The program change for the C64 is to make all the characters double-width. If you have a monitor for your C64, you may wish to redefine some characters back to their original PET or VIC configurations using the programmable character editor presented on page 32.)

Listing 2: **HPLOT for Commodore 64**

```
10 MX=39: MY=199: RW=MX+1: OG=1024: CM=55296
20 PRINT"□"
30 CC=4
100 FOR X=0 TO MX
110 Y1=X:X1=X
120 GOSUB 4000
130 NEXT X
900 GET T$: IF T$="" THEN 900
999 STOP
4000 REM HPLOT
4010 REM X1=0 TO MX      Y1=0 TO 8*MY-1
4020 IF HF THEN 4100
4030 DIM HF%(7): RESTORE: FOR I=0 TO 7: READ HF%(I): NEXT I: HF=-1
4100 Y1=MY-Y1
4110 PO=INT(Y1/8)*RW+X1
4120 CH=HF%(Y1AND7)
4130 POKE OG+PO,CH: POKE CM+PO.CC
4140 RETURN
4900 DATA 99,69,68,67.64.70,82,100
```

Line changes for VIC-20:

```
10 MX=21: MY=183: RW=MX+1: OG=4*(PEEK(36866)AND128)+64*
   (PEEK(36869)AND112)
15 CM=37888+4*(PEEK(36866)AND128)
```

Line changes for PET:

```
10 MX=39: MY=199: RW=40: OG=32768
4130 POKE OG+PO.CH
```

Suggested change for Commodore 64:

```
4900 DATA 119.69,68.67.64.70.82.111
```

Extra Horizontal Resolution

Listing 3 is a program similar to listing 2. This time the extra resolution is in the horizontal direction. The function shown is a scaled up natural log function. (Remember the log of 0 is undefined, so we have to skip 0.) Note how the extra horizontal resolution makes the curve smooth at the right edge. One precaution must be taken when using this routine. It is tempting to use Y1 as the index for the main FOR...NEXT loop, but you must use another variable instead. This is because the value of Y1 is changed within the subroutine.

Listing 3: VPLOT for Commodore 64

```
10 MX=319: MY=24: RW=40: OG=1024: CM=55296
20 PRINT"□"
30 CC=7
100 FOR Y=1 TO MY
110 X1=RW*2.5*LOG(Y)
115 Y1=Y
120 GOSUB 4000
130 NEXT Y
900 GETT$: IF T$="" THEN 900
999 STOP
4000 REM VPLOT
4010 REM Y1=0 TO MY      X1=0 TO MX
4020 IF VF THEN 4100
4030 DIM VF%(7): RESTORE: FOR I=0 TO 7: READ VF%(I): NEXT I: VF=-1
4100 Y1=MY-Y1
4110 PO=Y1*RW+INT(X1/8)
4120 CH=VF%(X1AND7)
4130 POKE OG+PO,CH: POKE CM+PO.CC
4140 RETURN
4900 DATA 101.84,71.66.93.72.89,103
```

Line changes for VIC-20:

```
10 MX=175: MY=22: RW=MX+1: OG=4*(PEEK(36866)AND128)+64*
   (PEEK(36869)AND112)
15 CM=37888+4*(PEEK(36866)AND128)
```

Line changes for PET:

```
10 MX=319: MY=24: RW=40: OG=32768
4130 POKE OG+PO,CH
```

50 × 80 Resolution

So far we have been able to increase the resolution in one direction at a time while maintaining only character resolution in the other direction. By using Commodore's full set of quarter-box characters, it is possible to double the resolution in both directions, creating 50 × 80 resolution on the PET and C64, 44 × 46 on the VIC-20, and 50 × 160 on 80-column machines. As with the bar-graph characters, some of the characters we need are hidden; they must be produced by reversing other characters. Fortunately, the PEEK/POKE code of a reversed character is simply the code for the corresponding normal character with 128 added.

I won't give a detailed description of the program, but a few bits of information might be useful in understanding it. The character codes are stored in the integer array QC%( ) in a logical order as if the four quarters of the character are the digits of a binary number. The lower left quadrant is the 1, the lower right quadrant the 2, the upper left quadrant the 4, and the upper right quadrant the 8. Thus, element 12 consists of the top two quarters. The plotting process consists of two separate processes: identifying the character position on the screen (line 5060) and determining the character quadrant we want to plot (line 5040). Once the character position is determined, we must decide if the character already there is one of our plotting characters (lines 5500-5550). If it is, the new point must be ORed into the old character so that information isn't lost.

Here's a programming challenge. Convert this routine to one that not only lets you set a point but also allows you to clear a point. (Hint: it involves the AND operation.)

Listing 4: QPLOT for Commodore 64

```
10 MX=79: MY=49: RW=40: OG=1024: CM=55296
20 CC=0: DIM QC%(15)
30 PRINT"□"
100 FOR X1=0 TO MX
110 Y=MY*SIN(X1*π/MX)
120 Y1=INT(Y+.5): GOSUB5000
130 NEXT X1
200 CC=7: FOR X1=0 TO MX
210 Y=MY*(SIN(X1*π/MX))↑2
220 Y1=INT(Y+.5): GOSUB5000
230 NEXT X1
900 GET T$: IF T$="" THEN 900
999 STOP
5000 REM QPLOT
5010 REM X1=0 TO MX   Y1=0 TO MY
5020 IF QF THEN 5040
5030 FOR I=0 TO 15: READ QC%(I): NEXT I: QF=-1
5040 XP=X1 AND 1: YP=Y1 AND 1: CN=(XP+1)*(3*YP+1)
5050 Y1=MY-Y1
5060 X0=INT(X1/2): Y0=INT(Y1/2): PO=Y0*RW+X0
5065 REM PRINTX1;XP;Y1;YP;CN:RETURN
5070 CO=PEEK(OG+PO): GOSUB 5500
5075 IF CC=(PEEK(CM+PO)AND15) THEN CN=(CO OR CN)
5080 POKE OG+PO,QC%(CN)
5090 POKE CM+PO,CC
5100 RETURN
5500 REM MATCH CHAR, RETURN QC% SUBSCRIPT
5510 FOR I=0 TO 15
5520 IF CO=QC%(I) THEN CO=I: I=15: NEXT: RETURN
5530 NEXT I
5540 CO=0
5550 RETURN
9000 DATA32.123.108,98,126,97.127,252,124,255,225,254,226,236,251,160
```

Line changes for VIC-20:

```
10 MX=43: MY=45: RW=22: OG=4*(PEEK(36866)AND128)+64*
   (PEEK(36869)AND112)
15 CM=37888+4*(PEEK(36866)AND128)
```

Line changes for PET:

```
10 MX=79: MY=49: RW=40: OG=32768
5075 CN=(CO OR CN)
```

Delete line 5090

Other applications of Commodore's built-in graphic characters include vertical bar graphs and a system involving the various line-intersection characters. No doubt you have already found uses for the ball and card-suit characters. The diagonal lines present some possibilities as do the rounded corners.

If the characters you need aren't there, you can always design your own using the programmable character editor on page 32. Future articles or columns will present some useful *sets* of programmable characters.

MICRO

MICRO™ Commodore Reviews

Product Name:	**64 Mail List**
Copy Protection:	Commodore 64 with 1541 Disk Drive and a printer
Price:	$34.95 disk; $29.95 cassette
Manufacturer:	Data Equipment Supply Corp. 8315 Firestone Blvd. Downey, CA 90241
Contact:	Computer Marketing Services Inc. 300 W. Marlton Pike Cherry Hill, NJ 08002

Description: *64 Mail List* is a mail-list program for the Commodore 64 that has quite a number of features. The program is menu driven and allows you to add to an existing file, correct the file you are working on, review, sort or print a file, merge two files, and create a Word-Pro file.

Pluses: The program is easy to understand. After choosing a main menu selection, you are given options pertaining to the function chosen; i.e., if PRINT PRESENT FILE is selected, you may choose either mailing-label printout or just names and phone numbers.

Minuses: The program is not well error-trapped. For zip code and area code, alphabetic input is accepted and may be far longer than any used in this country. If the printer is not on line when the print function is selected, an error is generated. At that point you cannot correct the error and continue but must start over, loosing all input not previously saved. Any error has the same effect.

Documentation: Brief but concise.

Skill level required: A beginner could use it if it were error-trapped.

Reviewer: Richard E. DeVore

Product Name:	**Casual Writer**
Equip. req'd:	Commodore VIC-20 with 8K minimum of memory, VIC Printer, and a VIC cassette unit
Price:	$29.95 plust shipping and handling
Manufacturer:	E.N. Publications R.D. 1, Box V Worden, IL 62097

Description: As a word-processing program for the VIC-20 the *Casual Writer* does allow you to type on the keyboard and print the results out *via* your printer. Being menu driven, the program is able to create a letter, print it, save it to tape, retrieve it from tape, delete a line, retype a line, insert a line, and view a letter on screen.

Pluses: *The Casual-Writer* does what it says it will do and it is easy to use.

Minuses: The program provides only upper case, does not allow the editing cursor movement and, therefore, does not let you change a word in a line. The line must be completely retyped.

Documentation: The program comes with five pages of documentation, which is sufficient to use the program.

Skill level required: May be used by a beginner.

Reviewer: Richard DeVore

Product Name:	**TYMAC Universal Tape Interface**
Equip. req'd:	For all Commodore computers
Price:	$49.95
Manufacturer:	Micro-Ware Distributing Inc. P.O. Box 113 Pompton Plains, NJ 07444

Description: Easily load and save programs in Commodore format on ordinary cassette recorders with this Cadillac of cassette interfaces. Although it costs more, it does more than competing units. I highly recommended it.

Pluses: This product solves the infamous parity problem between early and recent CBM Datasettes by including a parity switch. There are also three LED function indicators for motor on, read, and write, and there is a satisfying snap from a relay whenever a file is found or a load or save is completed. It also directly duplicates files from one cassette to another *without* loading into the computer.

Minuses: Like competing units, the interface plus cassette is bulkier than Commodore's Datasette alone. It also has a couple of more cables and requires attention to playback volume. No stated warranty. *(Editor's Note:* Manufacturer has informed us there is a 90-day warranty.)

Skill level required: Users should be able to plug in color-coded cables and set a few switches reliably prior to first use.

Documentation: The 8-page manual is ample for this product. Although the printing is tiny, the text is well-written and understandable. Included are simple demo programs and checkout ideas.

Reviewer: Jim Strasma

MICRO™ Commodore Software Catalog

Commodore Vadem to Develop Line of Micro-Wafer Products

Vadem, Inc. has announced its first offering for the low-end home computer market. The company's V-20 Expander unit for Commodore's VIC-20 home computer system is being marketed by Unitronics of Oakland, CA.

Measuring approximately 5x6x7 inches, the compact V-20 plugs into the VIC-20's cartridge extension slot. It includes a 10K-byte RAM memory expansion board, a 64K-byte data wafer and high-speed micro-wafer drive with read/write capability for mass data storage, a filing system, and Vadem's VWOS software.

Priced at about **$100** (retail), it will become available in the fourth quarter of 1983. For more information, contact Chikok Shing, Vadem, Inc., 3517 Ryder St., Santa Clara, CA 95051; (408)738-0571.

Connections for the Commodore

Krell's *Connections* is the most exciting development in educational computing since LOGO. CONNECTIONS offers children of all ages a new world of entertainment and intellectual challenge. Parents and educators will be gratified by the intriguing yet serious nature of CONNECTIONS.

CONNECTIONS is accompanied by an initial set of data bases (included free with the game system) that deal with geography, chemistry, mammals, mathematics, tools, and everyday objects. CONNECTIONS helps users to build their own data bases and to utilize the data bases created by others via the CONNECTIONS users group exchange program.

Price is **$99.95**. Contact Krell Software Corp., 1320 Stony Brook Rd., Stony Brook, NY 11790; (516)751-5139.

The Do It Yourself! Book for The VIC-20

Now for the first time ever there is a complete package especially designed for novices, users, and advanced programmers of the *VIC-20*. This package includes a step-by-step comprehensive tutorial with full explanations, program listings, exercises, programming methods, and many "trade-secrets" known before only to professionals — all in 25 easy-to-understand chapters. Some of the topics covered are hi-res and multi-color character design, special sound effects, binary-decimal-hex conversions, and a fast bubble sort.

For more information contact Softron, Inc., 2067 Broadway, Suite 27, New York, NY 10023; (212)490-0077.

MICRO

Commodore Hardware Catalog

Sound Shaper Creates New Instruments
for the Commodore 64

Sound Shaper has been developed to create new sounds from computer-generated instruments as well as music that resembles popular instruments such as the piano, xylophone, and banjo. At the touch of a single key users can test two million various combinations of sound produced by the synthesizer chip.

The Shaper's visuals include bar charts and digital displays of the sound being employed. In addition, the Sound Shaper allows for three independent "voices" to be adjusted separately for attack, decay, sustain levels and waveforms.

The Sound Shaper is available from most Commodore dealers or directly from Quality Computer, (213)501-4179.

Professional Word Processor
for the PET/CBM and Commodore 64

PaperClip was designed to overcome the limitations found on the word processors currently available on the market. It allows the author to create and edit text easily. Original text can be stored on a cassette on disk drive for future reference. Advanced features include horizontal scrolling, column manipulation, alphanumeric sorting, and printer support.
$10.00

Batteries Included
186 Queen st. W.
Toronto, Ontario,
Canada M5V 1Z1
(416)596-1405

Attention Hams and Shortwave Listeners!
for the Commodore 64 and VIC-20

Let your *Commodore 64* or *VIC-20* help you "listen in" on the world of Radioteletype and Morse code. **SWL RTTY/MORSE** converts your VIC-20 or C-64 computer into a low-cost RTTY and morse-decoding and display terminal. It allows you to receive 45-to 100-baud BAUDOT RTTY and 5- to 30-wpm morse code. It requires a TTL-compatible terminal unit such as the MFJ-1224/5 or the HRA Electronics TU-II. The package includes software on cassette, special user port connector, and complete instructions.

$19.95 plus $2.00 S&H

Kinetic Designs
1187 Dunbar Ct.
Orange Park, FL 32073

Numeric Keypad
for the Commodore-64 and VIC-20

Computer Place introduces **The CP Numeric Keypad.** The keypad is designed with top quality, low profile key switches for smooth, reliable, and low-cost data entry. It lets you zip through your numeric work sheet, input your numbers and figures comfortably, and quickly.

The setup is simple and the usage is comfortable.
$69.95

Computer Place
23914 Crenshaw Blvd.
Torrance, CA 90505
(213)325-4754

MICRO

Sound Software:
Jim: 1) VIC Music Composer, Thorn EMI, $39.95, 2) Synthesound, HES, $29.95, and 3) PIPER, Abacus, $19.95.
Dave: 1) PIPER, 2) Synthesound, and 3) VIC Music Composer.

Spreadsheats:
1) Practicalc, Micro Software Intl., $40, 2) Busicalc BC20, Skyles, $49, and 3) VICALC, UMI.

Database Managers:
1) Flex File 2.1, Webber, $110, 2) TOTL Label 2.1, TOTL, $20, 3) Research Assistant 2.0, TOTL, $30.

Book: *Mastering your VIC – 20*, MICRO, $19.95.

Commodore 64

Game:
Jim: 1) Jumpman, Epyx, $39.95, 2) Frogger, Sierra On-line, $34.95, and 3) Sword of Fargoal, Epyx, $29.95.
Dave: 1) Zork I, Infocom, $39.95, 2) Frogger, and 3) David's Midnight Magic, Broderbund, $34.95.

Educational:
Jim: 1) Tooth Invaders, Commodore, 2) Hey Diddle Diddle, Spinnaker, $29.95 and 3) Kinder Koncepts, Midwest Software.
Dave: 1) LOGO, Commodore, $59.95, 2) Turtle Graphics II, HES, $59.95, and 3) COCO, HES, $39.95.

(*Ed. note:* Dave is the author of Turtle Graphics II, but having reviewed it myself, I can add my own endorsement. *LW*)

Business Software:
1) MAS 64 Accounting Pkg., Info Designs, $80/module, 2) Cyber-Farmer 64, Cyberia, $195, and 3) The Businessman, Southern Solutions, $100.

Word Processors:
Jim: 1) Easy Script, Commodore, $50, 2) Paper Clip, Batteries Included, $125, and 3) Word Pro 3+/64, Pro Line, $80 (review in PET Vet, MICRO 61:10).
Dave: 1) Word Pro 3 with Spell, Professional, $99.95, 2) HES Writer 64, HES, $44.95, and 3) TOTL Text, TOTL, $40.

Graphics Packages:
Jim: 1) Doodle, City, $39.95, 2) Sprytebyter, Foxsoft, and 3) LOGO (listed #1 by Dave in the Educational category.)
Dave: 1) Sorcerer's Apprentice, Event Horizon, $49.95, 2) Sprite Master, Access, $34.95, and 3) Screen Graphics 64, Abacus, $24.95.

Sound Software:
Jim: 1) UltraBASIC 64, Abacus, $42.95, 2) Synthy 64, Abacus, $32.95 (review in PET Vet, MICRO 57:71), and 3) Music Composer, Commodore.
Dave: 1) Synthy 64, and 2) Synthesound-64, HES, $34.95.

Spreadsheet:
Jim: 1) Calc Result — Advanced, Handic, $149.95, 2) Calc Result — Easy, Handic, $79.95, and 3) Practicalc, Micro Software Intl., $49.95 tape, $54.95 disk.
Dave: 1) MultiPlan, Microsoft/HES, $99.95, and 2) Omni-Calc, HES, $49.95.

Database Managers:
Jim: 1) The Oracle, Batteries Included, $150, 2) Flex File 2.1, Webber, $110, and 3) Infomast, $100.
Dave mentions: Filing Assistant, Rainbow, $125.

Programmer's Aids and Utilities:
Jim: 1) Sys Res, Solidus, 2) Power 64, Pro Line, $99, (review in PET Vet, MICRO 65:12) and 3) Micromon, public domain, free.
Dave recommends the follows, though not in any particular order:
C-64 Link, Richvale/Computer Marketing, $185 (review in MICRO 59:110).
HESMON, HES, $39.95.
VIC Tree, Skyles, $89.95 w/o cable, $109.95 w/ cable.
Assemblers:
MAE, Eastern House, $99.95.
PAL 64, Pro-Line, $99.95.
Commodore Assembler 64, Commodore.

Languages and Compilers:
KMMM Pascal, Wilserv, $80 (review in PET Vet, MICRO 63:12).
64 FORTH, HES, $59.95.
C64 FORTH, Performance, $99.95 (review in PET Vet, MICRO 62:12).
DTL BASIC Compiler, CMD.

Communications Packages:
Jim: McTERM, Madison, $195, and STCP, Eastern House, $129.95.

PET:

Game:
1) Microthello, AB, $10, 2) Millipede, Nibbles & Bits, $15, and 3) The Dragon's Eye, Epyx, $29.95.

Educational Software:
1) COMAL 1.1, Instrutek, $495 in ROM, or COMAL User's Group, free (charge for mailing and disk), 2) Vanilla Pilot, Tamarack, $30, and 3) Commodore's Public Domain Disks, Commodore, $300.

Business Software:
1) BPI General Accounting, BPI, $400/module, 2) PET Speed 3.0, Small Systems, and 3) CMS Accounting System, Southern Solutions.

Word Processors:
1) Expanded Paper Clip, Batteries Included, $125, 2) Superscript, Precision, $250, 3) Word Pro 5+, Professional, $500.

Graphics Packages:
1) PIC Chip, Skyles, $80, 2) SuperGraphics, AB, $40, and 3) VIGIL, Abacus, $35. Jim highly recommends Visible Memory from MTU, but because of its cost ($495) and because it requires hardware (included), he left it off the list. It would have been first.

Sound Software:
1) 4 Part Harmony, AB, $40, 2) PIPER, Abacus, $25 and $30, and 3) Piano, Cursor, $6.

Spreadsheets:
1) Calc Result, Handic, $199, 2) VisiCalc, VisiCorp, $200, and 3) Colupad, Etcetera Intl., $150.

Database Managers:
1) Silicon Office, Bristol, $1000 and $1500, 2) The Administrator, Professional, $650, and 3) The Oracle, Batteries Included, $150.

Other Products Recommended:
Sys Res, Solidus.
POWER, Pro-Line.
Micromon, public domain, free.
McTerm, Madison, $195.
STCP, Eastern House, $129.95
Communicator, Amplify.
MAE, Eastern House, $99,95.
PAL, Pro-Line, $99.

Commodore Vender Addresses

Abacus Software
P.O. Box 7211
Grand Rapids, Mi 49510

AB Computers
215 Bethlehem Pike
Colmar, PA

Academy Software
Box 9403
San Rafael, CA

Access Software
925 East 900 South
Salt Lake City, UT 84105

Amplify, Inc.
2325 Macbride
Iowa City, IA 52240

Batteries Included
71 McCaul Street
Toronto, Ontario M5T 2X1
Canada

BPI
3423 Guadalupe
Austin, TX 78705

Bristol Software Factory
Kingston House
Grove Av., Queen Square
Bristol, England BS1 4QY

Canadian Micro Distributors
500 Steeles Avenue Milton
Ontario Canada L9T 3P7

Cardco, Inc.
3135 Bayberry
Wichita, KS 67226

City Software
735 W. Wisconsin Ave.
Milwaukee, WI 53233

The Code Works (Cursor)
Box 550, 5778 Hollister, Suite B
Goleta, CA 93116

COMAL Interest Group
505 Conklin Place
Madison, WI 53703

Commodore Business Machines
1200 Wilson Drive
West Chester, PA 19380

Computer Marketing Services, Inc.
300 W. Marlton Pike
Cherry Hill, NJ 08002

Creative Software
P.O. Box 4030
201 San Antonio Circle
Mountain View, CA 94040

Cyberia
2330 Lincoln Way
Ames, IA 50010

Eastern House
3239 Linda Drive
Winston-Salem, NC 27106

Epyx
Box 4247
Mountain View, CA 94040

Etcetera International
P.O. Box G
Apex, NC 27502

Event Horizon Software
Box 1327
New York, NY 10028

Fox Soft (Foxfire)
Box 507
Deer Park, TX 77536

Handic (see Computer Marketing)

Infocom
55 Wheeler Street
Cambridge, MA 02138

Info Designs
6905 Telegraph Rd.
Birmingham, MI 48010

Instrutek
DK-8700 Horsens
Christianholmsq.
Denmark 05-6111100

Madison Computers
1825 Monroe Street
Madison, WI 53711

Micro Software
50 Teed Drive
Randolph, MA 02368

Micro Technology Unlimited
P.O. Box 12106
2806 Hillsborough Street
Raleigh, NC 27605

Midwest Micro
311 West 72nd St.
Kansas City, MO 64114

Midwest Software
Box 214
Farmington, MI 48024

Nibbles and Bits
P.O. Box 2044
Orcutt, CA 93455

Nufekop
P.O. Box 156
Shady Cove, OR 97539

Performance Micro Products
770 Dedham Street, S-2
Canton, MA 02021

Precision Software
4 Park Terrace
Worcester Park, Surrey
England KT4 JZ

Professional Software
51 Fremont Street
Needham, MA 02194

Programmer's Institute
P.O. Box 3470
Chapel Hill, NC 27514

Pro-Line Software
775 The Queensway East
Mississauga, Ontario
Canada L4Y 4C5

Rainbow Computer
490 W. Lancaster Ave.
Frazer, PA 19355

Richvale Telecommunications
10610 Bayview Richmond Hill
Ontario Canada L4C 3N8

Sierra On-Line
Sierra On-Line Bldg.
Coarsegold, CA 93614

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

Small Systems Engineering
1056 Elwell Court
Palo Alto, CA 94303

Southern Solutions
P.O. Box P
McKinney, TX 75069

Spinnaker Software
215 First Street
Cambridge, MA 02142

Tamarack Software
Darby, MT 59829

Taylormade Software
P.O. Box 5574
Lincoln, NE 68505

Thorn EMI
Computer Software Development
1370 Avenue of the Americas
New York, NY 10019

TOTL Software
1555 Third Avenue
Walnut Creek, CA 94596

United Microware Inc.
3503-C Temple Avenue
Pomona, CA 91768

VisiCorp
1330 Bordeaux Drive
Sunnyvale, CA 94086

Webber Software
Box 9
Southeastern, PA 19349

Wilserv Industries
P.O. Box 456M
Bellmawr, NJ 08031

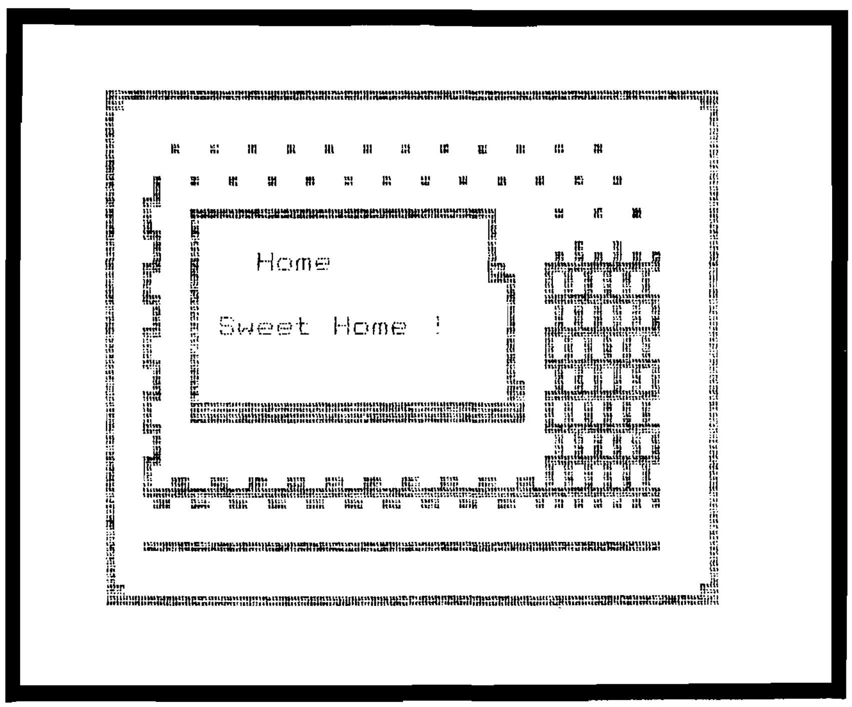
Introduction to TRS-80C Graphics

by Phil Daley

It's here! Finally Radio Shack has brought out a really powerful operating system for the CoCo. OS-9 has features that take the Color Computer out of the games arena and turn it into a top-end micro. Read all about this system in an article by Childress.

John Steiner reports on some new software and his BBS. He will have an in-depth review of BASIC09 as soon as possible.

The CoCo screen editor allows typing of the graphics characters on your screen to help in charcter graphics designs and screen displays. The binary file load on the CoCo is at least twice as fast as any of the other micros. The PRINT statement SAVE really demonstrates how to fool BASIC by modifying a program in memory.



CoCo SCREEN EDITOR

by Phil Daley

Important note! When typing the listing of the CoCo Screen Editor into your computer, be sure to type lines 0-3 exactly as shown but without any spaces. Failure to observe this precaution will cause a major crash when saving a screen. Also, be sure to save the program before running it as the program erases itself during the screen save.

Program Operation

The program first asks if you want to SAVE or LOAD a screen picture. Then an 'S' will ask if you want a Binary or Program save. The binary-save routine saves your picture with SAVEM to disk (add a 'C' to all the I/O commands for tape systems) and requires a 'LOADM' to retrieve the picture for use in another program. The program-save routine saves the picture in a BASIC program file, which can be RENUMbered and MERGEd into any other program. The binary-load routine asks for the filename and loads the screen into the editor placing the cursor in the top left corner.

The screen memory starts at 1024 and uses 512 memory locations. The numbers stored in screen memory are different from ASCII except for the upper case letters. The CoCo takes care of translating from CHR$ standard ASCII to the screen display codes.

The Screen Editor allows placement of characters (upper or lower case) and character graphic blocks (any color). There are several "control" keys used for editing and color changes. Here is a list of the commands:

Key	Command
Up arrow	Cursor up
Down arrow	Cursor down
Left arrow	Cursor left
Right arrow	Cursor right
Enter	Cursor to new line
Clear n	Set color n (n is 0 to 7) (Cursor is current color)
Clear up arrow	Clear screen
''	Up arrow
Shift 0	Lower case
&	Save current screen
Shift left arrow	Toggle from chars to blocks
$	Prints current screen

To return the screen to the original background color in case of a mistake, set the cursor color to the background color and in block mode type an 'F'; if the background is black, then in block mode type '0'. You should be able to create any kind of character picture that you want with eight character-block colors. When you are ready to save the picture, type '&' and the cursor will disappear. When the cursor returns the picture is saved to disk. If you specified 'P' for program save, you must save the picture by typing the SAVE command. DELete line 0 before saving or running the program.

Program Description

Lines 1-3 are dummy lines to create a place to store a set of strings containing the screen information. It is very important that these lines contain the correct number of characters as the screen information is POKEd to these program locations. Line 4 fills in the lower right corner of the screen. If you want to get fancy, modify the SAVE routine to POKE the correct value into the location in line 4 where the '128' is. Lines 5 and 6 end the saved portion of the program. The remainder of the program is deleted during the SAVE routine, so be sure to save the program before running it.

Line 30 is the subroutine to read the screen at the position V lines down from the top and H characters in from

(Continued on page 47)

This Christmas, Atari and Commodore* owners will go out of their minds.

It'll take about 30 seconds.

Because once you boot a Datamost game on your Atari home computer, or your Commodore 64, you'll come face to screen with the most mind-blasting games ever.

And what better time to go out of your mind than Christmas?

Our Music Will Have You Hearing Things.

Going out of your mind never sounded so good.

Because now our games have music. You heard right. Music. Original Music. Throughout. And scored just for our newest releases. You'll go nuts over it.

Our Graphics Will Have You Seeing Things.

You and everybody else.

Because our games are so great–how great are they?–they're so great you'll want to play them again and again and again. And then your friends will go bonkers over them.

And they'll want to play. And then your family will want to play. And then total strangers off the street will want to play and...

Mind-blowing arcade-quality action like this makes it one mad world, pal.

We'll Torment You Right From The Start.

No time to settle down and get comfy.

The tension begins building the moment you boot the game.

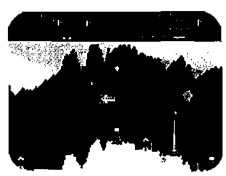
Terrific screen titles tease you with the game's objective, scenario, characters.

And while you wait, hand twitching over the Joystick, you'll find out what planet you're on.

What the fuss is all about.

Why you're involved. And perhaps of singular importance to you personally, how to keep from being obliterated. So get ready to get the most out of your mind.

By getting the most out of ours.



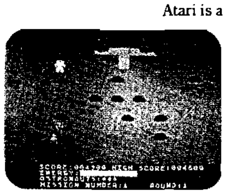
THE TAIL OF BETA LYRAE.™ Changes as you play. Unpredictable. Impossible to master. (No one has!)

DATAMOST™ INC

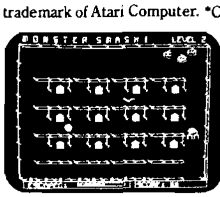
The most out of our minds.™

Datamost, Inc., 8943 Fullbright Ave., Chatsworth, CA 91311, (213) 709-1202

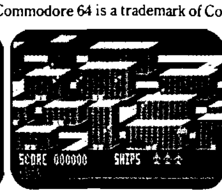
Atari is a trademark of Atari Computer. *Commodore 64 is a trademark of Commodore Business Machines, Inc. TM Registered Trademark of Datamost.



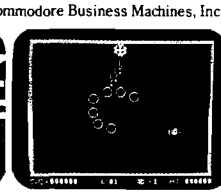
COSMIC TUNNELS.™ Four games in one. Four times the challenge! Incredible graphics.



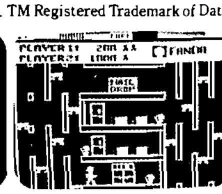
MONSTER SMASH.™ Deathly strategy. Mash the monsters! Let the visitors live.



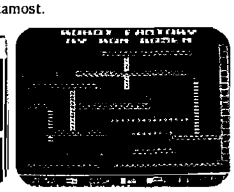
NIGHTRAIDERS.™ Strafe a city under siege with 3D angled selective firing!



ROUNDABOUT.™ Sharpens your shoot 'em up skills. 24 different game screens. Habit-forming!



COHEN'S TOWERS.™ You're the mailboy in a big city skyscraper. Work your way to the top.



MR. ROBOT.™ Screens scream with color, action and sound! Design your own screens, too.

the left. Line 50 is the subroutine to place a character on the screen with the same Vertical and Horizontal variables. Line 70 initializes the control-key variables. If you wish to customize the program, then this is the line to change.

Lines 90-120 update the cursor position, making sure that the cursor is within screen limits. Line 130 saves the current character in X0. Lines 140-150 print the cursor at the proper position. If you wish a different cursor character, change line 140.

Lines 160-170 get a new character from the keyboard and assign it to XN. Lines 195-240 check for cursor movement (control characters) and, if so, replace the old character on the screen and increment the cursor position.

Line 250 checks for the '&' and goes to the file-save routine if true. Line 260 looks for the CLear key and changes the current color. Line 270 sets and resets the graphic-character flag. If the flag is not set, then line 280 skips the graphics routine at lines 290-310. Lines 320-330 convert lower-case input into the proper values to POKE to the screen. Lines 340-350 POKE the character to the screen and update the current cursor position.

Lines 370-390 change the current color or clear the screen; lines 400-411 determine the type of save or load; line 420 loads a binary file; and lines 450-460 save a binary file, removing the cursor before saving.

Lines 460-599 contain the routine that POKEs the screen information into the program lines 1-3 at the beginning of the Screen Editor. I1 and I2 are then start and end addresses of the BASIC lines in memory. The subroutine at 600 actually does the storage. The '135' is a PRINT token, the '34' is the " (double quote), and the routine at 610-620 converts the lower case back to proper form for string storage. In line 660, the '34' is the ending quote and the '59' is a semicolon to prevent screen scrolling when the line is printed. These POKEs will work only if the beginning lines are typed accurately.

Line 590 deletes the rest of the program (also delete line 0 before saving) so that the resulting save is of the screen information. This short program can be RENUMbered and MERGEd for inclusion with other programs. Do not try to EDIT one of these lines since, when they are listed, they are untokenized, look messy, and are far in excess of allowable line length. In fact, the list routine doesn't even list the whole line, but it is there in memory just the same. This technique can be used to include any characters that cannot be entered from the keyboard into your program.

Line 700 restores the current character and reads the Model I conversion factors. This is necessary since the Model I used 2 × 3 block characters B & W and the CoCo uses 2 × 2 blocks in 8 colors. Lines 730-770 read and print the standard characters. Lines 790-810 perform the conversion to characters that appear somewhat similar on the printer. Obviously, they are not the same proportion vertically. Line 870 is the data to convert 2 × 2 into 2 × 3 graphic blocks. If you prefer a one to one conversion (2 × 2 into 2 × 2), use the data in line 880. The conversion is more exact, but the printout looks worse because of the space between screen lines. If you have only graphics characters, you can crunch them up by changing the "8" in line 720.

MICRO

```
   ******************************
   *                            *
   *     CoCo Screen Editor     *
   *       by Phil Daley        *
   *     Copyright © 1983 by    *
   *         MICRO Ink          *
   *       P.O. Box 6502        *
   *     Amherst, NH 03031      *
   *                            *
   ******************************

0 GOTO70
1 ::::::::::::::::::::::::::::::::::::::::
  ::::::::::::::::::::::::::::::::::::::::
  ::::::::::::::::::::::::::::::::::::::::
  ::::::::::::::::::::::::::::::::::::::::
  ::::::::::::::::::::::::::::::::::::::::
  ::::::::::::::::::::::::::::::::::::::::
  ::::::::
2 ::::::::::::::::::::::::::::::::::::::::
  ::::::::::::::::::::::::::::::::::::::::
  ::::::::::::::::::::::::::::::::::::::::
  ::::::::::::::::::::::::::::::::::::::::
  ::::::::::::::::::::::::::::::::::::::::
  ::::::::::::::::::::::::::::::::::::::::
  ::::::::
3 ::::::::::::::::::::::::::::::::::
4 POKE2047,128
5 IF INKEY$="" THEN 5
6 END
20 'READS SCREEN
30 PS=V*32+H:XO=PEEK(PS+1024):RETURN
40 'PRINTS NEW CHAR ON SCREEN
50 PS=V*32+H:POKE1024+PS,X1:RETURN
60 'INITIALIZE
70 DIM A(15):DN=10:CR=13:BK=8:FD=9:UP=94:
   ESC=38:H=0:V=0:CO=4:CL=12:CH=21
75 GOSUB 400
80 'MAIN LOOP
90 IF H<0 THEN H=31:V=V-1
100 IF H>31 THEN V=V+1:H=0
110 IF V>15 THEN V=0
120 IF V<0 THEN V=15
130 GOSUB 30
140 X1=131+CO*16
150 GOSUB 50
160 A$=INKEY$:IF A$="" THEN 160
170 XN=ASC(A$)
180 'CHECK FOR CURSOR MOVEMENT
190 IF XN=UP THEN X1=XO:GOSUB 50:V=V-1:GOTO 90
195 IF XN=34 THEN XN=94
200 IF XN>31 AND XN<>38 THEN 280
210 IF XN=CR THEN X1=XO:GOSUB 50:V=V+1:H=0:GOTO 90
220 IF XN=DN THEN X1=XO:GOSUB 50:V=V+1:GOTO 90
230 IF XN=BK THEN X1=XO:GOSUB 50:H=H-1:GOTO 90
240 IF XN=FD THEN X1=XO:GOSUB 50:H=H+1:GOTO 90
249 'CHECK FOR SPECIAL CHARS
250 IF XN=ESC THEN GOSUB 450:GOTO 90
255 IF XN=36 THEN GOSUB 700:GOTO 90
260 IF XN=CL THEN GOSUB 370:GOTO 140
269 'SET OR RESET GRAPHICS CHAR
270 IF XN=CH THEN FG = NOT FG: GOTO 90
280 IF NOT(FG) THEN 320
289 'CONVERT TO GRAPHICS
290 IF A$>"F"ORA$<"A"
    ANDA$>"9"ORA$<"0" THEN 160
300 XN=ASC(A$)-48+7*(A$>"9")
310 XN=128+CO*16+XN: GOTO 340
319 'CONVERT LOWER CASE
320 IF XN<64 THEN XN=XN+64: GOTO 340
330 IF XN>96 THEN XN=XN-96
339 'PRINT CHAR
340 X1=XN:GOSUB 50
350 H=H+1:GOTO 90
360 'CHANGE COLOR
370 A$=INKEY$:IF A$="" THEN 370
375 IF A$="↑" THEN CLSO
380 CO=VAL(A$):IF CO>7ORCO<0 THEN CO=5
390 RETURN
400 CLS:PRINT@64,"sAVE OR lOAD":INPUTA$
405 PRINT:PRINT"WHAT FILE NAME?":INPUTF$
410 IF LEFT$(A$,1)="L" THEN 420
411 PRINT:PRINT"bINARY OR pROGRAM?":
    INPUTG$: CLSO:GOTO 80
420 LOADM F$: GOTO 80
450 X1=XO:GOSUB 50
460 IF G$="B" THEN SAVEM F$,&H400,
    &H7FF,&HO: GOTO 80
470 SA=1024: I1=9742: I2=9989: GOSUB 600
480 I1=9995: I2=10242: GOSUB 600
490 I1=10248: I2=10274: GOSUB 600
500 CLS:PRINT@64,"IMMEDIATELY TYPE "
510 PRINT"SAVE "CHR$(34)F$CHR$(34)
520 PRINT"TO SAVE YOUR PICTURE"
530 DEL 10-9999
599 END
600 POKE I1,135
601 POKE I1+1,34
605 FORI=I1+2 TO I2-2
610 T=PEEK(SA):IF T<32 THEN
    T=T+96: GOTO 650
620 IF T>95 AND T<128 THEN T=T-64
650 POKE I,T: SA=SA+1: NEXT
660 POKE I2-1,34:POKE I2,59: RETURN
700 X1=XO:GOSUB 50:FOR I=0 TO 15:
    READ A(I):NEXT
710 PRINT#-2:PRINT#-2
720 PRINT#-2,CHR$(27)ACHR$(8)
730 FORI=1024 TO 1535
740 J=PEEK(I)
750 IF J>127 THEN 790
760 IF J<32 THEN J=J+96: GOTO 820
770 IF J>95 THEN J=J-64
780 GOTO 820
790 IF J<160 THEN J=J+16: GOTO 790
800 IF J>175 THEN J=J-16: GOTO 800
810 J=J-160:J=A(J)
820 PRINT#-2,CHR$(J);
830 IF (I+1)/32 = INT((I+1)/32) THEN PRINT#-2
840 NEXT
850 PRINT#-2:PRINT#-2
860 RETURN
870 DATA 160,168,164,172,162,170,166,174,
         161,169,165,173,163,171,167,175
880 REM DATA 160,192,176,208,162,202,190,
         222,161,205,181,221,163,207,191,223
```

OS-9: A NEW OPERATING SYSTEM FOR THE COLOR COMPUTER

by Stephen Childress

This discussion on OS-9 covers I/O and multi-programming commands, OS-9's reportoire of languages, and end-user software now on the market. But before we get too far, let's look at the typical hardware needed to run OS-9:

1. A microcomputer offering the Motorola MC6809 microprocessor. This includes the TRS-80C. The 6809 is often integral as in the SS50 Bus computers, but may also be an add-on as in the case of the Apple II.
2. A primary terminal using ASCII, sometimes with a serial interface. There are OS-9 configurations using memory-mapped video boards as well, including the 80-column boards for the Apple.
3. A boot disk. A good minimum storage size for an OS-9 system used for software developement or serious applications software is 320Kb, on two dirves, with five- or eight-inch floppies as desired. Hard disk(s) may also be present, but floppies are needed initially to load and boot OS-9. OS-9 has essentially no limit on the number of drives and each drive may contain 16 Megabytes. The "system" drive may be selected at any time and may be different for each user.
4. To use the multi-programming (tasking) feature, a periodic interrupt causation is needed. Often this is a time/counter chip with or without a calendar chip. In the case of the Apple II which lacks either, one of two schemes apply; A 60Hz signal connected to the cassette audio input jack, obtained from an IC pin which has the display video vertical sync, or, lacking this, the 6502 processor simulates the 60Hz as best it can, and it interrupts the 6809 and OS-9. The periodic interrupt for OS-9 may be obtained by several means, and the OS-9 "CLOCK" diver module may be altered accordingly.

Many OS-9 systems contain about 4Kb of "KERNAL" code in EPROMs so that the system may boot from most any device. Neither the CoCo nor the Apple II version require these EPROMs and the ROMs on the disk controller boot OS-9 from the system disk. The only other hardware essential is, of course, an adequate share of RAM. Except when OS-9 is used as the OS for a dedicated controller application, the RAM size is at least 56Kb.

Though we will not discuss it, Microware also offers a version of OS-9 called Level II, which supports larger memory sizes, including the 1Mb SS50 machines. This OS-9 is essentially equivalent from any one user's viewpoint, but of course the larger memory makes a vast difference in the number of simultaneous jobs or users that may be supported. Rumor has it that an OS-9 for the 68000-based systems is nearing completion by Microware. Also, GIMIX (1137 W. 37th Pl, Chicago, IL, 60609) has designed a 6809 CPU board that surpasses any 8-bit micro I've seen, it has memory-to-memory DMA to speed data shuffles and has most all of the system integrity checking hardware found in larger minicomputers, e.g., I/O instruction traps for programs not so entitled, and protection from a wayward program botching up other simultaneously running jobs. I've heard of other OS-9 work in progress but cannot comment yet.

Up and Running

After OS-9 is booted from floppy or hard disk, the first program executed is the "SHELL," a term coined to depict a layered software approach, i.e., the layers of skin found in an onion. SHELL's purpose is to serve as a command line interpreter (CLI), akin to the CLI found in most any computer system. (For those of you familiar with the UNIX Shell, OS-9's Shell is quite similar except for the absence of the elegant, built-in programming language.) SHELL sends the "OS-9:"

prompt to the primary terminal, TERM, and awaits a command from TERM's keyboard.

Usually there are certain functions to be performed each time the system is booted, for example, "loading" (fixing into memory) software modules for other terminals, disks, printers, or even often-used utility programs. To ease this task, OS-9 automatically looks for a startup file on the boot disk, which, if present, will contain any desired commands that would have otherwise been typed in from the terminal. The startup file is merely a procedure file containing ASCII text exactly as it would flow from a keyboard. Procedure files may be activated at any time by simply typing the file name from the keyboard, OR by placing the name in yet another procedure file. They may be nested as desired to create batch jobs.

After booting, set the date and time which, via the periodic clock interrupt, is maintained by OS-9. If a hardware calendar is present, this would be done automatically by a command within the startup file. Time-keeping is important since all OS-9 files are time and date annotated.

During startup, or at any time for that matter, device support software modules may be loaded from disk and attached by merely a simple keyboard command or from within a procedure file. This can even be done from within a program in BASIC! And conversely, if some device will not be needed for a while, drivers may be unloaded to free up memory space for other work. Remember the position-independent code story? Well, drivers, like any other OS-9 program module, are loaded on command into memory at whatever address region is uncommitted at the moment. This memory management scheme works quite well and is the cornerstone of OS-9's attractiveness for the small computer user.

The OS-9/User Intercom — SHELL

The Shell is the program in which most of the user/OS-9 conversation takes place. Adhering to the guidelines for OS-9 programs, Shell is position independent and reentrant; therefore many processes or users may simultaneously execute Shell, sharing its code. Let's take a look at Shell's main capabilities:

1. Accept commands to run programs in variuos languages. Shell commands may come from a keyboard or other peripheral or a disk *procedure* file. When invoked, a program is known as a *process* and is considered to be a *child* of the process which invoked it. In this case, the *parent* process is Shell.
2. Optionally, the new child process may be executed concurrently with the parent (in the background). Until memory space is exhausted, any number of processes may be created.
3. By default, the child process inherits the parent's I/O devices for the standard input, output, and error output. The beauty of this scheme is that OS-9 programs are naturally insensitive to what I/O devices are to be used. This is easy to see if we consider the case for a disk file lister program which may be run from any terminal — it too would use the parent's devices. But consider running some language like BASIC. Thus, we don't need special versions of programs, each patched for different I/O cases. The big payoff comes when several processes (or users) share the same program's code, say BASIC, but each has unique I/O devices.
4. Redirect the standard input, output, and/or error output paths of a program to any peripheral or disk file. This is quite handy to give the child process alternate devices, say a disk or printer for its output or input.
5. Wait for the completion of a child process (program), if desired.
6. Run the various utilites that show what processes now exist, and optionally abort them or change their priority, in terms of competition for CPU time slices.
7. Establish pipelines between the I/O channels of processes — i.e., connect a process's output to another's input. This permits a collection of general program tools to be combined in many ways to solve differing problems. For example, pipeline a *list* to a *word separator* to a *sort* to a *word counter* to a *word comparison* and you have a spelling checker.

Figure 1: OS-9 File Structure

```
                          /D0 (drive 0, root directory)
Parts (directory)           cmds (directory)          sys (directory)
  somefile (file)             attrib (file)             password (file)
  tires (directory)           copy (file)               errmsg (file)
    item (file)               date   ...(etc)
    afile (file)              del
    ...(etc)                  dcheck
  motors (directory)          dir
    stuff (file)              display
    info (file)               dsave
    ...(etc)                  dump
  ...(etc)                    exbin
                            ...(etc, etc)
```

Each diskette or hard disk, in this case one located in drive "/D0", contains a root directory and any number of directories. Any directory may contain files or other directories. In this simple example, there are only three directories in the root directory for /D0. The "parts" directory contains one file and three directories. Large disks in OS9 typically contain dozens of directories and hundreds of files. Floppy disks typically hold a few directories and dozens of files.

Now let's look at some Shell commands you would type in, and what they would accomplish:

copy text file to standard output (term):
OS9: list filename

same, but output redirected to a printer named "p", " > " redirects the standard output:
OS9: list filename >/p

same, but run child "list" as a background" process:
OS9: list filename >>/p&

Assemble a program with hardcopy listing and simultaneously edit some other file, " >> " redirects the error output from "asm"'
OS9: asm filename l o = progname /p oopsfile& edit somefile

Run a text file modifier utility, "fixtext", with filea as the input instead of the keyboard; fileb is the output in stead of the CRT, and the options for fixtext are -lf: remove line feeds; +uc: make upper case; and +ll=80: force the line length to 80 columns.

```
OS9: fextext   filea -lf +uc +ll=80
        fileb
```

Run a utility which counts the words found within the standard input:

```
OS9: wordcount   filea
```

Now, a Shell command with pipelining:

```
OS9: fixtext +ll//   filea ! word-
      count   result
```

will send filea through fixtext and, with the +ll=1 option, causes all words to be sent one per line to the standard output. This is piped to wordcount which counts the words and sends the summary "nnn words" to the standard output which has been redirected to a disk file named result.

Lastly, let's assume that there is a disk file named "job" which contains:

```
-x
del workfile1
del workfile2
x
report   company/invoices  +from=
      1/82 to=12/82   workfile1
      sort   workfile1 +key=date
        workfile2
list workfile2 ! specialformat   /p&
```

The "-x" tells the Shell to ignore errors; "del" deletes two files but with the -x option, the job continues even if they do not exist. The "x" Shell command returns to normal "fatal error" aborting. Now the "report" command is, say, an application program that uses a data base (such as "invoices") which exists in disk directory "company". Report uses "+from..." as options and sends the result to workfileel. "Sort" keys on, say, the "date" field of the report, creating workfile2. Finally, "list" was told to send workfile2 to "specialformat," which might format the disk file according to rules needed for a particular printer device, "p", and does so in the background since the printer is slow.

Now this procedure "job" is invoked by the user from the keyboard:

```
OS9: job
   or
OS9: job&
```

to run it as a background process.

As you might imagine, the Shell and procedure files permit complex but routine work to be performed easily, and, with procedure files, by a person with little computer training. Note also that all of the OS-9 programs called out here were unaware of the details of I/O. This included the names of the disk files, the options to use in processing, and the user-terminal and peripherals for the instance.

This procedure file defines a job to be done.

Most often, OS-9 for a general purpose micro is configured to assume that the primary terminal, called "TERM", is present and the first SHELL program uses it for the human interface. There may be other terminals, printers, disks, or whatever, but OS-9 starts out knowing only of TERM and the boot disk, which is usually called "D0" When OS-9 has finished the startup procedure, the SHELL program prompts for a command from the user via the TERM keyboard. (The startup file, if so programmed, may cause a turnkey application program to run instead of the SHELL). From the OS-9: SHELL prompt, any program, language, or procedure file may be run. Included with OS-9 are several dozen useful utilities, some of which are shown in table 1. Pay particular note of the special SHELL operators such as ">" and "&"; we'll see how these are used in a moment.

As with CP/M, a utility or application may be added by merely placing that file on disk and using the file name to run the program. Parameters and options needed by that program are placed to the right of the file name. For example:

OS9: date t	run the date utility with the "t"ime option

***Figure 2:* Standard Set of OS-9 Utilities (excluding OS-9 level two, the assemblers, editors, and languages)**

attr	change a file's read, write, and execute permission; public/private
backup	make disk backup
binex	convert binary to s-record
build	create a simple text file
cmp	file comparison utility
cobbler	install current boot program
copy	copy verbatim, old to new file
date	show the current date and time
del	delete file(s) if so entitled
dcheck	verify file structures on a disk volume
dir	brief or comprehensive listing of file name directory
display	send hexadecimal bytes to a file or device
dsave	build procedure file to copy files in a directory
dsave	generate procedure file to copy files
dump	show a file in a hexadecimal and ASCII byte form
echo	echo input to output path
exbin	convert Motorla Mikbug "S" format to an OS-9 binary file
format	initialize any floppy disk
free	show a selected drive's free space
indent	show a report of module's type, revision, size, etc
kill	abort specified process (task)
link	link a module into memory or increment its use count
list	show an ASCII file's contents
load	fix a module in memory from disk
login	used by tsmon to log onto timesharing system with password
makdir	create a new sub-directory
mdir	show a directory of loaded modules (memory directory)
merge	create a merger of files
mfree	show used and unused memory size information
os9gen	make new OS9 boot file to optimize or customize
printerr	enable user-supplied english error messages
procs	show currently active and sleeping processes (programs)
pwd, pxd	show the currently selected default directories
rename	change a file's name
save	copy a (modified) module from memory to disk

83/06/02 12:34:56	is displayed
OS9: setime 83 6 2 12 34 56	runs the set time utility

Now whenever a program is run, it becomes a *sibling* of the program which invoked it. The invokER is called the *parent*, and the invokEE is called the *child* or *sibling*. Each time a program is run, a new *sibling process* is established and given a number. Without the "&" in the SHELL command line, the sibling process (program) must complete before the parent process resumes. This is known as "sequential" execution. In the above "setime" example, SHELL is the parent (SHELL issued the OS-9: prompt), and setime is the child. Since sequential execution was specified (no "&" was present), SHELL will become dormant until setime completes.

If the child process needs to be run concurrently instead of sequentially, the "&" is used.

Each time a program is run it always inherits three I/O channels from its parent. The channel or I/O path scheme is used in OS-9 and is similar to the File Control Block idea in CP/M, et al. But the I/O path idea requires a program to worry only about a simple, one-byte file number; the dirty work of manipulating file control material is entirely the job of OS-9's various file managers for disk and character I/O. This greatly simplifies OS-9 programs. The standard channels for any program are:

#0	standard input, a terminal's keyboard in many cases
#1	standard output, often the screen or hardcopy of a terminal
# 2	*standard error, often the same as #1*

In the "date t" example above, the date program inherited the invoker's device, which may have been the primary terminal. But if this command is entered:

OS9: date t >/p (where "/p refers to a printer

the ">" causes the standard output to be rerouted to device /p for the date program (the sibling). When date runs, its output goes to a printer instead of the usual place.

File Directories

Most readers have become familiar with the concept of *tree-structured* or *hierarchical* files by now. Recall that the idea is to allow for disk drive's media to have a main or "root" directory in which are the names of files and the names of other directories. And in any of those directories, other files and other directories may appear, etc. This results in a picture of a system of files which is like a corporate organizational chart, where the root is the president, the directories in the root are the departments, and each department has manager-directories and therein worker-files. Remember that any directory may have files in addition to other directories.

As disk capacities *v.s.* cost decline, the number of files per drive quickly gets to a point where this segregation of files is essential in order to prevent chaos with dozens or hundreds of files. But there is a more important benefit as well. A program may be written to create or use files with predefined names but be assured that these files are unique because of the currently selected directory. For example, consider the financial accounts of several companies on the same disk but in distinct directories; all may have an "invoice" file.

In OS-9, like UNIX, a default directory exists, which is changable according to the user's or program's wishes. When a program opens or creates a file, this directory is used, unless an override is explicitly stated. This gives rise to the notion of *path names* instead of file names. If "afile" is stated, it is a file name expected to reside in the current directory. But to refer to other directories, a pathname is used, which

setime	manually set the date and time
setpr	change a processes's execution priority (0-255)
sleep	suspend for "n" clock ticks
shell	(see below)
tee	copy standard input to multiple outputs
tmode	show or alter terminal's or port's controls: LF, NULLS, Case, special keys for break and abort; XON/XOFF chars, etc, etc.
tsmon	activate time sharing monitor for some terminal
unlink	remove a module from memory if no one else is using
verify	verify a module's integrity, header parity & CRC-24
xmode	examine or change device initialization mode
...	many other utilities exist; some for sale & some public. These include directory copy, sorted volume directory, disk file patcher, disassemblers, and of course, languages.

shell	invokable as desired from most languages or another shell

Shell's built-in commands and options:

chd	select desired default data file directory
chx	select desired default execution file directory
;	separates multiple commands within a single line
&	causes preceeding portion of command line to run concurrently (background) and the SHELL continues. Dozens of "sibling" programs are permitted, according to memory size.
!	establishes a UNIX-style pipeline of data out = data in, between two programs.
ex	discard the current shell and run a program NOT as a child process. Used only rarely when Shell is no longer needed.
>	redirect standard input to a file or device
<	redirect standard output to a file or device
>>	redirect error output to a file or device
w	wait for (any) child process to complete

#nn or #nnK give additional RAM workspace to the program being run is a series of file names, separated with slashes "/" to show the parent/child relationship. In OS-9, a pathname that begins with a "/" is referring to the root or "main" directory on a device. For example, "/D0" is the root directory on drive D0, which is viewed via the command: OS-9: dir /d0. Now, presume that in this listing we saw a file name "parts" and, as disclosed by "dir", we noted that "parts" was a directory, not a data file. To see the files within "parts" we type: OS-9: dir /d0/parts, and we see yet another set of files and possibly other directories.

The default directory is selected by the Shell "chd" (change data directory) command: OS-9: chd pathname . If the current directory were "/d0" (which it is just after booting), then "chd /d0/parts" makes "parts" the default directory. Now a "dir" command will show the files within "/d0/parts", and the "/d0" will not be needed for files in the now-selected directory "parts". If the command "list filename" is now entered, that file must exist in "parts", and likewise, any program that does file opens or creates will do so in the "parts" directory. Now if inside "parts" there were yet another directory "tires", we would view its contents by "OS-9: dir tires, since "parts" contains tires".

The point is, again, that a program or, yes the users, may work with a resonably small number of file names at one time and be unaware that these files are in fact, perhaps one of many similiar sets of files. Files may be organized into directories according to most any criteria, say companies, projects, tasks, programming languages, etc. Directories may be created at any time, restricted only by the media capacity. The only limitation in OS-9 is that a file (or directory) cannot be spread onto two disk drives. This "multiple-volume" feature is found in some of the large computer systems, but is not a serious problem for micro's unless you've got data base sizes like insurance companies and Uncle Sam to contend with.

OS-9 does not support "alias" file names. With these, a file name may be created for an empty file which refers to a file in some other directory. Alias names are moderately useful, but the lack thereof is not serious.

We've not said so directly, but the pathname scheme applies universally to files for the Shell and also to files accessed by programs in any of the OS-9 programming languages. So that the user and programs may better ignore the parent/child and device residence of files, OS-9 provides for "." and "..", which are special directory names. Within any directory, ".." means the current directory's parent directory and "." means the current directory. Using again our "/d0/parts/tires" setup: "chd /d0/parts/tires" sets the default directory to "tires". Now "dir.." gives the files in "parts", and "list ../afile" opens and lists "afile" in the parent directory "parts". I'll bet you can now figure out what "chd .." and "list../../filex" do! The latter shows how one can most always ignore which disk drive is carring the files in use; this is a great advantage in that programs which open files may do so without sensitivity to what drive (or indeed, directory) is being used.

Execution Directories

So far we've discussed the directories for data file, making no mention of where executable files reside on the disk(s). Whenever a program is run, from a Shell command file, a procedure file, or by a programmed statement, OS-9 first looks for that program name in the "current execution" directory. After booting, that directory is "/d0/cmds", or, the "usual commands directory." Let's say that device "/d0" is in fact a floppy disk and, after getting started, we wish to switch everything to a hard disk named "/h0". This could be done with the following commands (which would likely be in a procedure file such as the standard startup file):

load h0driver	get the hard disk interface driver into memory
load h0	and get the 1st drive's descriptor module
chx x/h0/cmds	switch execution directory
chd /h0	switch data directory
setime /t2 /t2	get dte/time via terminal t2
shell /t2 /t2 /t2&	give a shell to terminal t2

Let's assume that this startup file was run during booting so the terminal used was the default term. The first four commands setup to use the hard disk; the *setime* prompts for and gets the date and time via terminal t2; finally, the procedure file runs the Shell with terminal t2 for I/O. Now since the chx and chd appeared in the procedure file ahead of the Shell command, t2 will inherit the hard disk for its default directories. After the Shell command is done, the procedure file is complete so the *invoking terminal (term)* receives control once again, and terminal t2 is also running.

Now each command entered at t2 will use the /h0/cmds directory to locate these programs. If however, the

***Figure 3:* Hardware manufacturers whose machines have OS-9 capabilities**

Radio Shack
One Tandy Center
Fortworth, Texas

GIMIX
1337 W. 37th Place
Chicago, Illinois 60609

Smoke Signal
31336 Via Colinas
Westlake Village, California 91362

Southwest Technical Products
219 W. Rhapsody
San Antonio, Texas 78216

Hazelwood Computer Systems
907 E. Terra
O'Fallon, Missouri

AAA Chicago Computer Center
120 Chestnut Lane
Wheeling, Illinois 60090

Stellation Two (Apple OS9)
The Lobero Building, PO 2342
Santa Barbara, California 93120

Those manufacturers who do not routinely advertise products have been excluded from this list. Several offer single-board computers in the under $1000 category though these are intended for OEMs or serious hobbists and have not been included. Your best source is to contact Microware Systems and inquire about supporting hardware and software for their OS-9.

program is in some other directory, the default directory may be overridden just as data file names may be. The command:

OS9: /h0/afile"

will attempt to execute a file located in the root directory rather than the usual /h0/cmds directory. Likewise, any disk or directory may be specified.

Program Module Validation

When a command line such as:

OS9: filename

is entered, the first action is to look for that file in the current EXECUTION directory. If found, it is loaded into memory wherever space exists (all code is position independent in OS-9). If that name is not found, perhaps it is a textual procedure file, so OS-9 looks in the current DTA directory for the file. If found here, it is treated as a series of Shell commands, until end-of-file is found. If it is found in neither directory, a "file not found' error occurs. What if the file was indeed text, not code? And further, what if the code is not machine language but instead BASIC or "p" code? No problem. All executable files for OS-9 have a header, automatically created by the compilers and assemblers, in which appears the code size, RAM size (minimum), module integrity checkword, and other pertinent data. With this, OS-9 can avoid confusing files of text with executable files, determine if any interpreter (e.g. BASIC) is required to run the module, and allocate code-space and RAM workspace from the memory pool. So no matter if the module is written in assembly language, BASIC, or Pascal, OS-9 will set up the execution situation without the user having to worry in what language the program was written.

Universal Input/Output

One of the most valuable features of OS-9's UNIX legacy is the elegantly simple I/O scheme. Here we're speaking of non-disk I/O, to printers, terminals, tapes, special interfaces, or whatever. In summary, the OS-9 scheme for such devices is to let each device simulate a seqential disk file. By that we mean exactly a "file"; it may be opened, written to, read from, and closed. Anything OS-9 does with disk files may be done for peripheral devices. In fact, a "seek" for random I/O or a "create file" may be done for devices (though to no effect). Consider this OS-9 BASIC program fragment:

```
INPUT "Say hi on what?"; FILENAME
CREATE #PATHNUM, FILENAME
PRINT FILENAME "HELLO"
CLOSE #PATHNUM
```

This program works regardless if the user enters the name of a disk file or the name of a peripheral, so long as the peripheral is writeable.

Device names always begin with a "/", like root directories do and may be invented as needed according to the user's system configuration. For example:

/term	is most always the primary terminal
/t1	is often the second CRT or hardcopy terminal
/P	is often the high speed printer
/p1	might be a letter quality printer
/modem1	could be a 300 baud modem
/modem2	maybe 1200 baud
/a2d0	might be analog-to-digital #0

Adding devices is quite simple with OS-9, and there is no limit on the number of disk or peripheral devices connected. Device support software may be brought in at any time...there is no need to patch or reconfigure OS-9 itself to add or remove a device, including disks. This is in stark contrast to the terribly messy and complex means needed for other systems. In fact, any user or program is able to laod and unload device support software dynamically at will, so that the peripherals may come and go as needed (which saves memory space). Again, OS-9 is usually interrupt-driven by its devices. For non-interrupting devices, polling may be used instead of interrupts, perhaps using the periodic clock interrupt to cause a device handler to check a device many times per second. A device handler may "go to sleep" for a while and be resumed later — a feature useful for certain device types. (The Apple OS-9 system, for example, uses the 6502 to do all non-disk I/O concurrently, and the 6502 interrupts the 6809 card in the Apple for character I/O and clock service.) To dramatize this scheme, consider the examples shown in inset 1.

Inset 1

OS9: load modeml	device support for a modem
OS9: dialup 800/555-1212 /modeml	send phone number
OS9: converse /modeml /modeml	session with a distant system
OS9: copy /modeml afile	get data from remote
OS9: list /modeml bfile	get textfile from remote
OS9: list bfile /modeml	send textfile to modem
OS9: list bfile ! aprotocol /modeml /modeml	pipe through a protocol to modem
OS9: dir /d0/jim /modeml	send directory to modem
OS9: unlink modeml	remove now unneeded software

(Where "modeml" is the name of an OS9 software module, and "/modeml" is the pathname of the device.)

OS-9 Time-Sharing Subsystem

Not found in most any small computer is the fairly complete time-sharing system for OS-9. It is, in fact, a very small amount of code, since all of the basics for time sharing are already present in OS-9 (interrupts, multiple users, multiple processes, multiple and protected files). Suppose that we have several CRT and/or hard copy terminals attached via, say, serial ports. To activate these, we would need to merely run some program with its input/output redirected to the desired terminal, and do so as a concurrent process. If the terminal was, say "/t2", the most simple means would be:

OS9: load t2	bring device support into memory
OS9: shell /t2 /t2 /t2&	give t2 a shell; run concurrently ("&")

This gives the user at t2 a Shell and unrestricted access to the system. What does this cost, in terms of memory? Well, the module "t2" is typically less than 256 bytes, and Shell's code is already present so a second incarnation of Shell costs only a new data area of some 512 bytes. Now the t2 user is being supported as was the original, boot-time terminal term. Perhaps the user of t2 is indeed the person who was using term, but t2 is a fancier terminal or located in the family room or shop, distant from term. Ready for t3 yet?

Suppose that the t2 user was indeed connected via modem, and we wish to restrict his access to the system's files, and further, he must log on with an approved password. Enter the time-sharing monitor:

(Continued on next page)

OS9: tsmon /t2& startup timesharing on /t2, concurrently

Now when a person on the t2 port of the system hits return, he will get a message from tsmon:

OS9 Time Sharing System, date time, etc
User name?
Password:

When tsmon gets "user name," it looks for that in a file "/d0/sys/password." If not found, the user is reprompted up to three times. If the user name is valid, a password is found in the file for that usr and it too must be entered (secret; no echo). If both are correct, tsmon uses the password file to login the user. For each user-name, the password file contains:

1. Password
2. User number (to restrict files by "owner number")
3. Process priority (0-255, to control "CPU-time hogging")
4. Name of the user's initial data directory
5. name of the user's initial execution directory
6. name of the user's initial program to execute (typ. Shell)

Given these items, the newly logged-on person's privileges may be carefully controlled. For example, if the initial program is not "shell," the alternate program could be a turnkey application program, for example, a bulletin board system or a business program. The program may be written on any of the OS-9 languages. The initial data and program directories may be set to disallow a user from changing to other directories to which he/she is not entitled. The time-sharing system is not bullet proof, but for trusted employees or casual, computer-club use, it is indeed practical on even small 64Kb machines.

Summary

How does OS-9 differ from CP/M, MS-DOS 1.0, FLEX and the other circa 1975 DOS's? Here are the key points. OS-9:

1. Uses modules of code and data which are all position-independent regardless of the language in which a module is written.
2. Programs use only the code and data space needed so that many programs may simultaneously coexist and run concurrently.
3. Totally eliminates conflicts in memory address contention among programs running sequentially or concurrently.
4. Allows the user and/or other programs to ignore specific address regions in which programs are run at any given time. This is true for the programmer as well as the user.
5. Provides multi-user and multiprogramming without awkward or restricting schemes as in other systems, and does so with the memory usage efficiency needed for small computers.
6. Provides tree-structure file directories and device-transparancy.

Figure 4: Survey of OS-9 Languages and Applications (as of June 1983)

The following is a list of programming languages with which the author has had either direct or indirect exposure. (The manufacturer's addresses are given in figure 5.) You may also wish to contact the hardware manufacturers for information regarding the software they independently offer. Apologies now to those software and/or hardware houses not mentioned...There are many others, mainly with smaller packages, but space does not permit us to list them all. You can find them advertising in the pages of *MICRO* magazine and *68 Micro Journal*. Many magazines covering the TRS Color Computer are beginning to show OS-9 advertisements.

Assembler **Microware Systems Corp**

This assembler supports the module scheme of OS-9, lacks macros, but does have conditionals. It is quite fast. It does not support relocatable code with a linking loader, thus many languages suppliers supply their own assembler. Non OS-9 code may be written for other 6809 machines.

Assembler **Conejo Computer Products**

Compatible with the Microware assembler; also supports the semantics of the Technical Systems Consultants (TSC) assembler for FLEX-9; Macros are supported beyond that of the FLEX assembler; has support for 680x source code. Is somewhat slower than the Microware assembler though. Also supports non OS-9 code. Price is quite attractive.

BASIC09 **Microware Systems Corp.**

To call this "BASIC" is a travesty! BASIC09 is a highly structured language as is OS-9 itself. It is an interactive compiler with a built-in editor and debugger for creating and working with its Pascal-like code. It will accept most old-style BASIC code with optional line numbers, but really shines when the local procedures and data structures are used. Most users proclaim it to be far and away the best language they've used, especially in terms of the speed at which programs may be written and proved correct. To date, it is the only BASIC available for OS-9, perhaps because to out-do it would be a huge effort.

Pascal **Microware Systems Corp.**

An extensive implementation, featuring automatic overlays to permit quite large Pascal programs to be compiled and run. Compared to others, the compiler is large and slow. The version seen had little debugging support.

Pascal **Omegasoft**

A "lean-machine" Pascal, with many extensions to support real time and I/O work. An interactive, symbolic debugger is included, along with a library, relocating assembler, and a linking loader to simplify program mergers. Disk storage requirements are modest.

COBOL **Microware Systems**

This is an OS-9 adaptaton of "CIS COBOL", a popular language within the micro community. The popular "Forms" package is available for larger OS-9 systems.

C **Introl Corporation**

This C compiler includes all of the "K&R" standard excepting a few compiler pragmas and initilializers which are easily worked around. An assembler, linker, and library are included. With the rapidly increasing popularity of C, the C compilers will become more important. This one supports virtually all of the features of the big-machine compilers, which will benefit the user since C has brought a greater degree of portability of programs than any language to date. Goodly sized disks are essential though; bare-bones mini-floppies will be over taxed.

Applications Software (Partial listing)

STYLOGRAPH — **Great Plains Computer**
A very useful word processor that gives you a "what you see is what you get" editor, unlike many others that are two-steps, edit/format/re-edit, etc. Also supports fancy printers, including legal printwheels, etc. Easy to use. Has spelling checker and form letter options.

Accounting — **Specialty Electronics**
Accounts receivable, payable, invoicing, inventory, etc; the staff of life for small business accounting.

Spreadsheet — **Computer Systems Center**
Supposedly better than "you-know-who"! Allows for generalized data files, shared among application programs.

Spreadsheet and DBMS — **Smoke Signal Computers (see hardware)**
Brand nw, this extensive package is called "Total Management Planning (TMP)," and includes a spreadsheet, DMBS, personal filing, and others. Unique in that it promises excellent training with audio and video cassette tape training beyond user manuals.

Text Editor — **Microware Systems Corp.**
A line-oriented editor that has macros and large file capabilities.

Text Editor — **Conejo Computer Products**
This editor is similiar to the well known UCSD Pascal screen editor, requiring only a $500 class CRT terminal.

"various" — **Frank Hogg Lab**
This company has advertised numerous OS-9 packages, mainly in the area of system software and utilities. Write for a catalog.

"various" — **South East Media**
Like "Instant Software," is a subdivision of a magazine, in this case, *68 Micro Journal.*

"various" — **Alford & Associates**
They offer a screen editor/word processor, speech synthesizer, proof reader, and utilities.

Figure 5: Software Manufacturers' Addresses (Partial Listing)

1. Microware Systems Corp.
(OS-9 Author)
5835 Grand Ave.
Des Moines, Iowa 50304

2. OmegaSoft
POB 842
Camarillo, CA 93010

3. Introl Corporation
647 W. Virginia St.
Milwaukee, WI 43204

4. Conejo Computer Products
31220 La Baya Drive, Suite 110
Westlake Village, CA 91362

5. Great Plains Computer Co.
POB 916
Idaho Falls, ID 83402

6. Specialty Electronics
POB 541
2110 W. Willow
Enid, OK 73701

7. Comuter Systems Center
13461 Olive Blvd.
Chesterfield, MO 63017

8. Frank Hogg Lab
The Regency Tower, Suite 215
770 James St.
Syracuse, NY 13203

9. South East Media
POB 794
Hixson, TN 37343

10. 68 Micro Journal
5900 Cassandra Smith
POB 849
Hixson, TN 37343

7. Permits a program to support multiple users, multiple instances of itself, and I/O device transparency easily. This yields many benefits, one subtle one being ideal portability of programs among various OS-9, one for small 64Kb machines, and another for larger machines with up to 1Mb of memory.

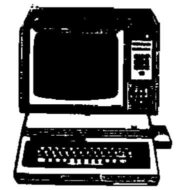
The Future

As far as ability to tackle most any job, OS-9 certainly qualifies. The available language reportoire (figure 4) shows that the tools are there to develop almost anything imaginable. There is a modest but reasonably complete set of applications software available, and it is growing more rapidly of late, but it does remain sparse at this time. From the available software products, it appears as though OS-9, to date, is primarily used by OEMs as a development tool for industrial controllers and by universities for computer science courses. The recent appearance of more business software, and especially the new GIMIX "multi-user" CPU board, will likely spur on more OS-9 applications in general computing. The Smoke Signal OS-9 computer was the subject of and did very well in some very impressive business software benchmarks (as opposed to the simplistic sieve benchmarks). The Stellation Two Apple II OS-9 system is interesting in that the Apple computer owner may exchange the severely limited DOS 3.3 for a more modern and capable OS.

Perhaps the UNIX, XENIX, etc, explosion will spin off to OS-9 via the C language, since those programs will run under OS-9 without great alteration. It is also rumored that a 68000 version of OS-9 is soon to come; this would offer many small computer users an inexpensive UNIX-like system with a processor even more capable that the 6809. Meanwhile, OS-9, either Level One for 64Kb machines, or Level Two for larger ones, is available to those with an interest in the benefits of modern software approaches. The author would appreciate hearing from those readers who would like to learn more about OS-9 from additional articles which cruise through OS-9-land.

MICRO

MICRO™ CoCo Bits



John Steiner

At this early point, I have not yet seen either the new Color Computer 2, or the 64K CoCo, but by the time you read this, I hope to be experimenting with one. Though there is little difference with the new computers. I am anxious to try out the new keyboard.

Last month I commented that a new keyboard would be standard on the 64K computer. However, if you have been in a Radio Shack store, you have probably already found out that the Color Computer 2 sports the deluxe keyboard as well. The new keyboard lists for $39.95 and is available as an upgrade kit as part number 26-3016. I have been told the kit will fit on early Cocos, but haven't been able to verify this for myself. I have a reservation about this because in October, 1982, Radio Shack made a change in the keyboard connector. Early versions had a multi-pin male connector on the circuit board, while the new connector is a female edge connector that mates with a plastic flexible circuit board that connects to the keyboard matrix. Other suppliers of add-on keyboards offer an adapter to fit either type; I hope Radio Shack will do the same.

More on the 1.1 Disk ROM

I have been experimenting with the new 1.1 disk ROM this month, and I have found that there seems to be no difference in the two ROMs when working with BASIC. I have been told that the new ROM will not format a diskette unless the computer has the complete series of new ROMs (BASIC 1.2 and Extended BASIC 1.1), so I commented last month that the new disk ROM wouldn't work with the old BASIC ROMs. I did have trouble formatting diskettes, but I have been able to format using the new ROM so evidently my information was incorrect. I had even tried the format command, and it hadn't worked, but I traced that incident to a couple of faulty disks.

Just yesterday Terry Jensen of J&M told me that an improperly timed drive unit will not format a disk correctly. All other disk operations will seem to work normally if the timing is only slightly off. I will be checking the timing on my drives this week, and by next month should be able to clear up the mystery of disk formatting on the new ROM.

Dakota Database

Last month I commented on a new bulletin board service that was started in the Fargo area. Unfortunately we have to change the phone number that

CoCo Bits *(continued)*

the unit is connected to, and since this column is being written during the telephone strike, I cannot supply the new number. Next month I will provide the correct access number. If you have tried to call and got a disconnect message, I am sorry. By this time directory assistance should have the number, which is listed under J. Steiner.

I am intrigued by the popularity of the BBS in less than two months. As I write this, we have had over 700 calls into the system, and now have over thirty regular users. If you are interested in starting a BBS and would like assistance, let me know. Be prepared to spend up to an hour daily updating and maintaining the system. If you want it customized, all you need is a knowledge of BASIC.

OS-9 Released

The long-awaited release of OS-9 has occurred. The powerful DOS supports multi-tasking and includes a full-featured editor/assembler. It should be available through Radio Shack soon. Costing only $69.95, this Unix-like system will make a wide range of sophisticated software available for the CoCo. I have ordered a copy, and will report on it as soon as it arrives.

Radio Shack's first offering for OS-9 is BASIC-09, billed as an interactive compiler. This Pascal-like implementation of BASIC allows named procedures and has a large complement of loop constructs. In addition to FOR-NEXT, it has WHILE-WEND and DO-UNTIL, among others. I have worked with BASIC-09 from Microware, and I am hoping the CoCo implementation is complete. I will have more on BASIC-09 as it becomes available.

Tandy Releases Educational Software

Radio Shack is going all out for the educational market with CoCo software. From "Telling Time With Donald" to "Goofy Covers Government," the characters of Walt Disney are employed to assist the learning process. Age range of the program series is five to fifteen, and all use high-resolution graphics, sound, and music. All programs require Extended BASIC. I am looking forward to evaluating these programs for their educational content.

Games Galore

It isn't often that I discuss games or associated software for MICRO, but I have a couple of comments on some newly release games. If there is one thing CoCo doesn't lack, its quantity of high quality games. Computerware of Encinitas, CA, has sent me copies of four new releases, MOONHOPPER, BLOCHEAD, NERBLE FORCE, and MOROCCO GRAND PRIX. Since I am not much into games, I turned these over to my resident CoCo games expert, my ten year old son, Josh. His favorite is BLOCHEAD, which is an adaptation of the arcade game Q-BERT. MOONHOPPER, another arcade type is his second favorite. This game has you controlling a car roving across the lunar surface. The object is to travel to a distant port while evading enemy missiles and craters in the lunar surface.

MICRO

MICRO™

TRS-80C Hardware Catalog

SS-50 Buss

The **DM-64 Dynamic Memory Board** uses low-cost 8116/4116 Dynamic RAM. All memory chips supplied on assembled boards are prime 8116-200 ns. The board operates like static with totally transparent refresh and can be used with 16K to 64K memory. Each board is solder-masked and silk-screened and fits into an 8 7/8-inch by 5 3/4-inch space. Power requirements are 1 amp at +5 VDC, 200 ma. at +12 VDC, and 50 ma. at −5 VDC.
$219.00 (64K) for an Assembled board
$50.00 for an Bare board

Southeastern Micro
Systems, Inc.,
1080 Iris Drive, Suite 202
Conyers, GA 30207
(404)922-1620

The Voice Pak — A CoCo Voice Synthesizer

The Voice Pak is a complete phoneme-based voice system that uses the famous VOTRAX SCO1 synthesizer chip in a cartridge-style pak. It provides an unlimited vocabulary with automatic or user-supplied inflection, a variable voice-level adjustment, plus four programmable levels of pitch. With a single line of code, THE VOICE PAK adds speech to any BASIC program in minutes. The system comes complete with a user instruction manual, software cassette with demo programs, text-to-speech scanner, and a Word Manager that constructs custom-user dictionaries. The unit is fully assembled, tested, and ready to plug in and talk.

THE VOICE PAK now allows any prompting application in education, speech therapy, games, robotics or security to be cost effective.
$69.95 plus $3.00 shipping

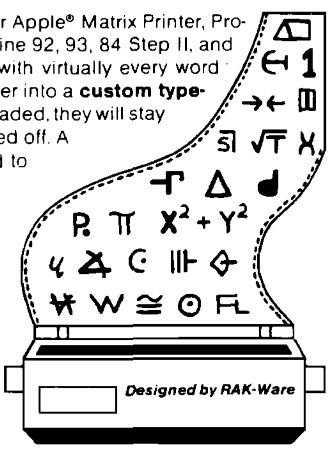
Spectrum Projects
93-15 86th Drive
Woodhaven, NY 11421
(212)441-2807

C-Line Suspension File Disk Holder

New No. 63759 Suspension File Folder by C-Line Products, Inc., securely holds and protects two 5¼" disks and two index cards. The non-glare vinyl holder provides excellent visibility and complete protection. Two index cards permit ready identification and accounting for location of disks when removed from files. Fits into any standard hanging file system (letter size only).

No. 63759 Suspension File Holders are packed 10 to a box.
$22.50

C-Line Products, Inc.
P.O. Box 1278
1530 E. Birchwood
Des Plaines, IL 60018



MICRO™
TRS-80C Software Catalog

Tax Command Prepares Individual Tax Returns

Tax Command, a Federal Income Tax calculation program, (for the Color Computer, Apple, Commodore 64, VIC-20, and Atari) is now in its second year of publication. It provides a line-by-line method of calculating income tax for federal tax forms, including form 1040, income averaging, Schedule A — itemized deductions (including medical), capital gains and loses, and contains all tax tables for every filing status. On computers with over 48K, Tax Command includes numerous other schedules as well.

Tax Command is easy to use. It does all mathematical calculations automatically, contains built-in tax tables that calculate your tax refund or payment, and tells when to income average. While the program does not print on the actual form, in most versions it does print (or list if you do not have a printer) each entry needed on your tax form.

Suggested retail is **$24.95** and is available for dealer and distributor sales and also for individual sales.

An Integrated Accounting System for the TRS-80 Color Computer

Universal Data Research, Inc. announces a completely integrated accounting system for the TRS-80 Color Computer using the Flex: operating system. The accounting system includes five main accounting packages: Accounts Payable, Accounts Receivable, General Ledger, Inventory, and Payroll.

All of the accounting packages are menu driven and easy to use. They create and maintain their own data files and are designed to eliminate unnecessary re-entry of information. Numerous reports can be generated from menu choices in each program or through the UDRI Data-Base Manager. All of the programs were written through the Data-Base Manager and can be easily modified to fit any company's specific needs.

To run the UDRI integrated accounting system, you need 64K RAM, Flex* operating system, TSC Extended BASIC, and at least two disk drives. A printer is recommended for report generation. The UDRI Integrated Accounting System may be modified with the UDRI| Data Base Managers or customized by UDRI.

Prices and additional information may be obtained from Universal Data Research, Inc., Cynthia D. Wilson, Software Mkt. Mgr., 2457 Wehrle Drive, Buffalo, NY 14221; (716) 631-3011 or its authorized distributors.

MICRO

MICRO™ TRS-80C Reviews

Product Name: **NEWTALK**
Equip. req'd: Color Computer with 16K memory
Price: $20.00 tape or disk
Manufacturer: Star Kits
P. O. Box 209-N
Mt. Kisco, NY 10549

Description: *NEWTALK* reads memory beginning at a specified address and speaks out the starting address and hex contents. If the operator presses ",", the next address and contents are spoken; "." causes only the contents of the next address to be enunciated, while "/" rereads the current location. The "voice" is husky and somewhat noisy but intelligible. The program is written in position-independent code, so it can be relocated as needed.

Pluses: This program is useful for checking hand-entered code without a helper; and it is a starting point for computer speech experiments. It can also be a "gee whiz" to awe your friends.

Minuses: It would be convenient to have a control key to step backward in memory; the major flaw is that "zero" is enunciated "oh".

Documentation: A nine-page manual of excellent quality is furnished. A full listing of the speech driver is given (the "speech" is recorder digital sound representation), and a limited amount of explanation for modification and experimentation is given.

Skill level required: None.

Reviewer: Ralph Tenny

Product Name: **RS-232/C Expansion Cable**
Equip. req'd: TRS-80/TDP 100 Color Computer
Price: $19.95 + $2.00 shipping
Manufacturer: Spectrum Projects
93-15 86 Drive
Woodhaven, NY 11421

Description: The Spectrum Projects RS-232 expansion cable allows the connection of two serial devices to the single output cable connector on the Color Computer. Essentially, the cable connector is a Y-type parallel connection with a male DIN plug that fits the CoCo and two female DIN plugs that will fit a standard CoCo RS-232 device.

Pluses: The cable eliminates disconnecting and reconnecting a printer to hook up a modem, for example. In addition, the extra cable length is usually helpful, especially in cases where the modem or printer must be located on a different table from the computer. The device is convenient and works well when the equipment will accept it (see below).

Minuses: The cable will not work with all combinations of printer/modem/other device hookup. It does work with the LP VII, VII, DMP series and the standard Radio Shack modems. My particular setup caused problems. My printer (an Epson MX-80) refused to operate with my Mura MM-100 modem connected. Evidently the Muro modem grounds the status line to the printer when it is in the off position, causing the computer to refuse to send data to the printer.

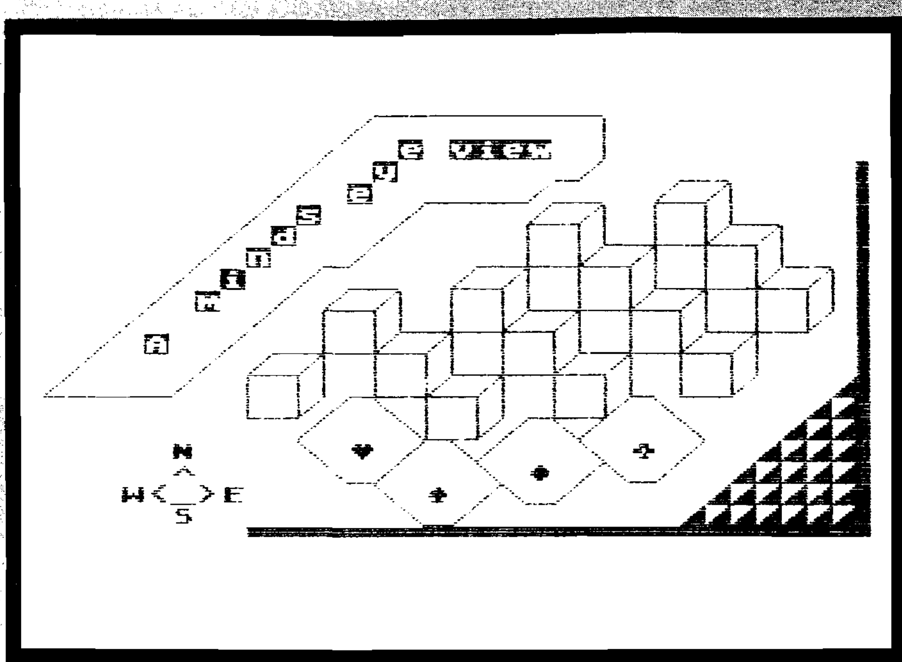
Documentation: None supplied or required.

Skill level required: The cable is easy to install and will work with no problems assumming the two RS-232 units are compatible.

Reviewer: John Steiner

Introduction to Atari Graphics

by Phil Daley



This month we will explore the Atari's character graphics capabilities with several programs by our Atari experts — Paul Swanson and Tom Marshall. Tom has written a screen editor that disables some of the Atari's features that can cause problems when drawing on the GRAPHICS0 screen. He has also added a SAVE and LOAD functions so that you can use these screens made with the editor in your own programs. In addition, a drawing package has been included, that redefines some of the characters to enable easy construction of pictures using several line drawing routines. Tom also wrote a program to define your own character sets and graphics characters with routines to SAVE and LOAD these from disk or cassette. With the help of Paul, Tom presents a screen dump (in the screen editor program) for the Epson printer to enable printing of any possible character that the Atari can create.

Paul has written a program to complement his MODE10 series that can convert MODE10 paintings into MODE0 pictures, saving a great deal of storage space. The program designs its own multi-color characters that imitate the MODE10 screens by causing the computer to interpret the screen as a GTIA screen instead of the equivalent of a MODE8 screen. This means that, for a screen that can be converted, a saving of 6K of storage space is possible. Using a 16K screen storage area will allow 8 full screens to be stored at one time for rapid display shifting or simple animation. MICRO

Atari Screen Editor

by Tom Marshall

The Atari comes with cursor-oriented editing built in. The text can be changed and moved about to and from any part of the screen with the CTRL and arrow keys. Add to this Atari's graphics characters and you can make quite a variety of screen displays.

There are a couple of problems that you should know about when using the editor. First, the normal functions of the editor can get in the way. For instance, it is easy to accidentally clear the screen with a single keystroke, and hitting return will not only cause the screen to scroll, but will cause BASIC to try to enter in a line thereby giving an error message. Using the memo pad (*via* the BYE command) helps a little by, in effect, disabling BASIC, but things are still not quite right.

Secondly, there is no way to save the screen for later use. Granted, the character displays are limited to the monochromatic keystrokes available on the Atari, but wouldn't it be nice to have a screen image saved on disk or cassette for all time?

The following program takes care of all the shortcomings of the internal screen editor.

In this program, all except the destructive abilities of the internal editor are allowed. The Clear key won't clear the screen and the Return key won't cause the screen to scroll. The rest of the abilities are more or less intact. When you wish to save or load in a file, hit the ESC key. Hitting ESC sounds a buzzer and then puts the program into the command mode. The options in this mode are ESC, 'S', 'L', and 'C'. Hitting the ESC key again simply puts you back into the screen-editing mode. Hitting the letter 'C' clears the screen (it was put in the command section to decrease the possibility of accidental clearing).

Hitting the 'S' key will allow you to save your screen to tape, disk or printer. When you hit 'S' the program should pause for several seconds while arrows appear on the left edge of the screen. Don't worry, they will not destroy your picture; instead, the program uses this time to read what is on the screen. When it is done (when the arrows reach the bottom), the screen will clear and you will be asked for a filename. Specify 'C:' if you wish to save the screen to cassette or 'D:filename' if you wish to save it to disk.

Next the program will ask if you wish to save the screen as straight data (screen file) or in the form of a BASIC program. If you select 'B', the file will be composed of about 24 line numbers (the program will prompt you for the starting line number and the increment to make the following ones) followed by PRINT statements and the various screen lines in quotes. It is for this reason that double quotes are not allowed.

Hitting the 'L' key will allow you to load in a screen already on disk. Note that, as with the saving subroutine, you will be asked whether or not the file you wish to load is in BASIC format. This is to enable you to load in and re-edit a BASIC format file you have already entered. *Do not mix up these file types!* It could be disastrous if you attempt to load from a supposedly BASIC format file that is really a screen file or *vice-versa.* When the loading is complete you are returned to the editing portion of the program.

Program Description

Line 20 calls the initialization subroutine at lines 30000 to 30090. This routine opens the keyboard for reading and reads several short machine-language routines into strings. Then it uses the block move routine to move the ROM character set into RAM. The data statements are read and POKEd into the first 16 character positions for the draw line characters.

Lines 100 to 132 are the screen-writing routine. It was placed near the top for execution speed. Line 100 gets the ASCII value for a character typed. It also throws away the 'CLEAR' code. Line 102 goes to the draw routines for processing there. If it is a normal character then control is passed to the normal printing routine at line 120. If the value is 34 (ESC key), then control is passed to the command routine at line 200. If the value is 155 (Return key), then control is passed to the return routine at line 130. When the routine is complete, the program will then go back and get another character.

The routine at lines 10000 through 10330 saves the screen. It can be saved in one of two ways — in screen form or in BASIC form. If it is to be saved in screen form, then the screen is simply output as 24 lines of 40 characters. If it is to be saved in BASIC form, then the 24 lines are output preceded by line numbers and print statements and surrounded by quotes. This creates a ready-to-run BASIC program output in ASCII that can be ENTERed into the Atari later. This routine is very useful, especially for enthusiasts who often incorporate screen images into their programs.

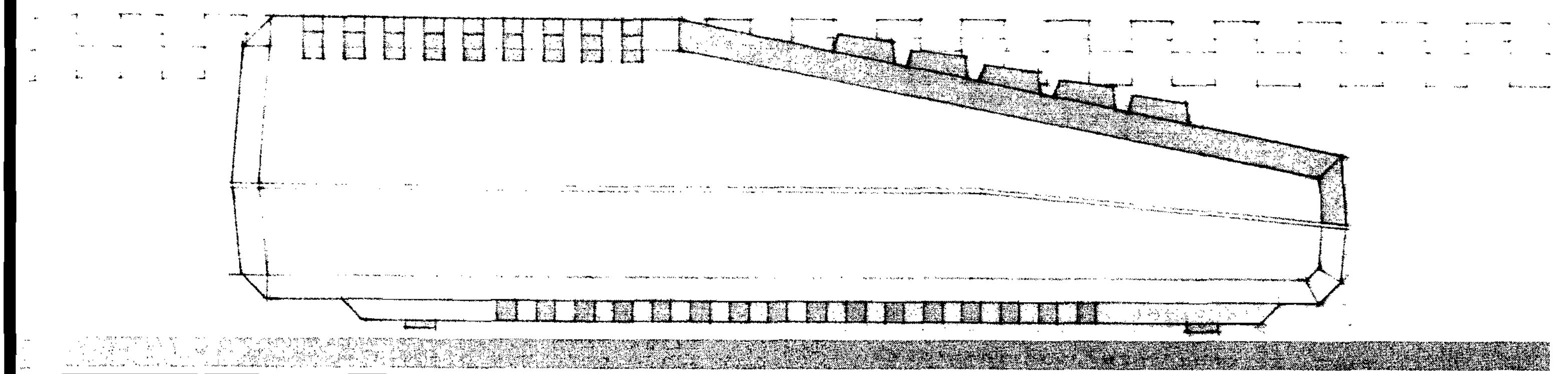
The routine at lines 11000 through 11360 loads in a screen. As with saving screens, loading can be done in one of two ways — in screen form or in BASIC form. If screen form is specified, then the loading is done line by line, 40 characters at a time. If BASIC form is specified, then the program loads in the file, line by line stripping off the line numbers and quotes. Again, do not mix up these two modes when loading as the loading might crash. If this should happen, simply re-run the program and specify the right type of file. The routine at lines 21400-21770 is the draw package which is explained in the general information section.

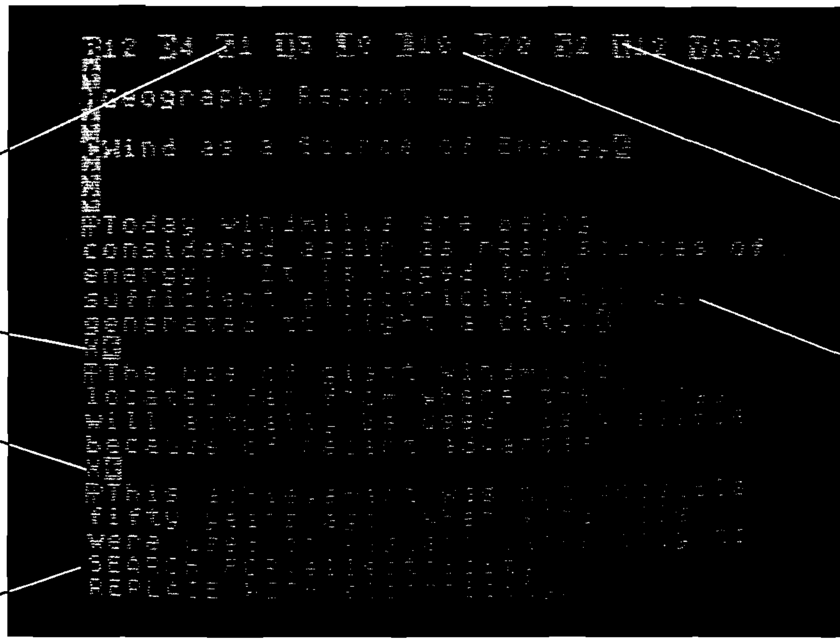
(Continued on next page)

Listing 1

```
10 DIM S$(960),F$(15),TEMP$(40)
11 REM DIM X$(1):A=ADR(X$):B=INT(A
   /256+1)*256:DIM F$(B-A-1)
12 DIM BUF1$(8),BUF2$(8)
13 DIM OUT$(640),LINE$(120)
14 A1=ADR(BUF1$):A2=ADR(BUF2$)
20 GOSUB 30000
30 SCREEN=PEEK(88)+PEEK(89)*256:DF
   =0:MODE=0:PEN=1
40 POSITION 0,0:? "→←";
100 GET #1,C:IF C=125 THEN 100
102 GOSUB 21400
105 IF C=34 THEN 100
110 IF C=27 THEN 200
115 IF C=155 THEN 130
116 IF PEEK(85)=39 THEN POKE 85,0
120 IF PEEK(85)=39 AND PEEK(84)=23
    THEN POSITION 0,0
122 PRINT CHR$(C);:GOTO 100
130 IF PEEK(84)=23 THEN POSITION 0
    ,0:PRINT "→←";:GOTO 100
132 PRINT :GOTO 100
200 PRINT "▣";
210 GET #1,C:IF C>128 THEN C=C-128
220 IF CHR$(C)="␛" THEN 100
222 IF CHR$(C)="S" THEN 10000
224 IF CHR$(C)="L" THEN 11000
226 IF CHR$(C)="C" THEN 12000
228 GOTO 210
500 OPEN #4,8,0,"P:":PRINT #4;"␛A"
    ;CHR$(8)
520 POSITION 0,0:PRINT S$(1,959);
530 ? #4:? #4
540 FOR I0=1 TO 960 STEP 40
580 FOR J=0 TO 39:LINE$(LEN(LINE$)
    +1)=S$(J+I0,J+I0):NEXT J
590 CC=0:OC=1:OUT$="♥":OUT$(640)="
    ♥":OUT$(2)=OUT$
600 CC=CC+1:FLAG=0:CNO=ASC(LINE$(C
    C)):IF CNO=193 THEN BUF2$="▮a▮
    ▮Aa▮":A1=ADR(BUF2$):POKE 1663
    ,1:GOTO 660
602 IF CNO>127 THEN CNO=CNO-128:FL
    AG=1
610 IF CNO<32 THEN CNO=CNO+64:GOTO
     630
620 IF CNO<96 THEN CNO=CNO-32
630 CNO=CNO*8+START:A1=CNO:BUF2$="
    ♥♥♥♥♥♥♥♥"
640 POKE 1663,1-FLAG
650 A=USR(ADR("hh▮Oh▮Nh▮Oh▮P)♥▮R ♥
    IN▮)/P▮)▮aQN ♥/HQ/PISR9W/▮♥*1
    P*▮P HQ/PS)/fRERPd♦"),A1,A2)
660 FOR I=1 TO 8:OUT$(OC,OC)=BUF2$
    (I,I):OC=OC+1:OUT$(OC,OC)=BUF2
    $(I,I):OC=OC+1:NEXT I:IF OC<64
    0 THEN 600
700 A=LEN(OUT$):HI=INT(A/256):LO=A
    -HI*256:? #4;"␛L";CHR$(LO);CHR
    $(HI);OUT$
710 OUT$="♥":OUT$(640)="♥":OUT$(2)
    =OUT$:OC=1
740 LINE$="":NEXT I0
750 CLOSE #4:POKE 752,0:POSITION 0
    ,0:? "→←";:GOTO 100
10000 REM ---Save Routine---
10002 POKE 752,1:PRINT "→←";:S$=""
    :COLOR ASC("→")
10008 FOR Y=0 TO 23
10010 FOR X=0 TO 39
10020 LOCATE X,Y,C:S$(LEN(S$)+1)=C
    HR$(C)
10030 NEXT X:PLOT 0,Y
10040 NEXT Y
10100 REM -Which type?-
10110 POSITION 0,7:PRINT "  Enter
    ▣rint ▣creen or ▣asic save"
10120 GET #1,C
10130 IF CHR$(C)="S" THEN GOSUB FI
    LE:GOTO 10200
10140 IF CHR$(C)="B" THEN GOSUB FI
    LE:GOTO 10300
10150 IF CHR$(C)="P" THEN 500
10160 GOTO 10120
10200 REM -Screen Save-
10210 FOR PTR=1 TO 960 STEP 40
10220 PRINT #2;S$(PTR,PTR+39)
10230 NEXT PTR:CLOSE #2
10240 PRINT :PRINT "  Saved to ";F
    $;" in screen form.":END
10300 REM -BASIC Save-
10310 POSITION 0,9:PRINT "▣Line #:
    Start, Increment...";:INPUT L
    N,IN
10312 IF LN+IN*39>32767 THEN PRINT
    "?Line numbers will exceed 32
    767.":GOTO 10310
10320 E=39:FOR PTR=1 TO 960 STEP 4
    0
10321 IF PTR=921 THEN E=38
10322 PRINT #2;LN;" PRINT ";CHR$(3
    4);S$(PTR,PTR+E);CHR$(34);";"
10324 LN=LN+IN:NEXT PTR:CLOSE #2
10330 PRINT :PRINT "  Saved to ";F
    $;" in BASIC form.":END
11000 REM ---Load Routine---
11010 S$=""
11100 REM -WHICH TYPE?-
11102 ? "Enter file.":INPUT F$:OPE
    N #2,4,0,F$
11104 POSITION 0,7
11110 PRINT "  ...Enter ▣CREEN or
    ▣ASIC load..."
11120 GET #1,C
11122 IF CHR$(C)="S" THEN 11200
11124 IF CHR$(C)="B" THEN 11300
11126 GOTO 11120
11200 REM ---Screen Load---
11210 FOR PTR=1 TO 960 STEP 40
11220 INPUT #2;TEMP$
11230 S$(LEN(S$)+1)=TEMP$
11240 NEXT PTR:CLOSE #2
11250 POSITION 0,0:PRINT S$(1,959)
    ;:POSITION 0,0:PRINT "→←";:GOT
    O 100
11300 REM ---BASIC LOAD---
11310 FOR PTR=1 TO 960 STEP 40
11320 INPUT #2;TEMP$
11330 FOR Q=1 TO LEN(TEMP$):IF TEM
    P$(Q,Q)<>CHR$(34) THEN NEXT Q
11340 S$(LEN(S$)+1)=TEMP$(Q+1,Q+40
    )
11350 NEXT PTR:CLOSE #2
11360 POSITION 0,0:PRINT S$(1,959)
    ;:POSITION 0,0:PRINT "→←";:GOT
    O 100
12000 PRINT "K";:GOTO 100
13000 POSITION 0,5:PRINT "  Enter
    filename......";:INPUT F$
13002 OPEN #2,8,0,F$
13005 RETURN
21400 REM **Process Draw Commands
21402 A$=CHR$(C)
21405 IF C=34 AND MODE=1 THEN MODE
    =0:RETURN
21410 IF C=34 AND MODE=0 THEN MODE
    =1:RETURN
21412 IF C=253 AND DF=0 THEN DF=1:
    SS$="":C=34:RETURN
21414 IF C=253 AND DF=1 THEN DF=0:
    C=34:RETURN
21416 IF DF=1 THEN SS$(LEN(SS$)+1)
    =A$:IF LEN(SS$)>=50 THEN DF=0
21420 IF MODE=0 THEN RETURN
21450 REM ****Decode keys: Y=N,U=N
    E,J=E,M=SE,N=S,B=SW,G=W, and T
    =NW
21451 IF A$="Y" THEN 21481
21452 IF A$="U" THEN 21482
21453 IF A$="J" THEN 21483
21454 IF A$="M" THEN 21484
21455 IF A$="N" THEN 21485
21456 IF A$="B" THEN 21486
21457 IF A$="G" THEN 21487
21458 IF A$="T" THEN 21488
21459 IF A$="H" THEN PEN=-PEN:GOTO
    21660
21460 IF A$="S" THEN 21700:REM Squ
    are
21461 IF A$="Z" THEN 21710:REM Bri
    ck
21462 IF A$="L" THEN 21720:REM Len
    gth
21463 IF A$="W" THEN 21730:REM Wid
    th
21464 IF A$="R" THEN 21740:REM Rec
    tangle
21465 IF A$="D" THEN 21750:REM Dec
    ision
21466 IF A$="C" THEN 21760:REM Can
    ned String
21470 RETURN
21480 REM ***Routines to write 8 d
    irections
21481 GOSUB 21602:GOTO 21501
21482 GOSUB 21608:GOSUB 21502:GOTO
    21501
21483 GOSUB 21601:GOTO 21502
21484 GOSUB 21503:GOSUB 21604:GOTO
    21502
21485 GOSUB 21503:GOTO 21602
21486 GOSUB 21504:GOSUB 21503:GOTO
    21608
21487 GOSUB 21504:GOTO 21601
21488 GOSUB 21504:GOSUB 21604:GOTO
    21501
21500 REM **Move cursor
21501 ? "↑";:GOTO 21510
21502 ? "→";:GOTO 21510
21503 ? "↓";:GOTO 21510
21504 ? "←";
21510 RETURN
21600 REM **Calculate Draw Pattern
    s
21601 BP=1:GOTO 21610
21602 BP=2:GOTO 21610
21604 BP=4:GOTO 21610
21608 BP=8
21610 SX=0:X1=SCREEN+PEEK(84)*40+P
    EEK(85)
21615 S=PEEK(X1):IF S>127 THEN S=S
    -128
21620 IF S<16 THEN SX=S
21630 IF PEN=1 THEN S=USR(O16,SX,B
    P)
21640 IF PEN=-1 THEN S=USR(A16,SX,
    255-BP)
21650 IF PEEK(85)=39 AND PEEK(84)=
    23 THEN POSITION 0,0
21652 IF PEEK(85)=39 THEN POKE 85,
    0
21655 ? CHR$(S+32);"←";
21660 C=34:RETURN
21700 REM ***Square
21701 FOR I=1 TO LN:GOSUB 21481:NE
    XT I
21702 FOR I=1 TO LN:GOSUB 21483:NE
    XT I
21703 FOR I=1 TO LN:GOSUB 21485:NE
    XT I
21704 FOR I=1 TO LN:GOSUB 21487:NE
    XT I
```

(continued)





You can't find a friendlier, more powerful word processor at twice the price. New AtariWriter™ Under $100.

Now you can do multi-featured word processing at home, simply. At a family budget price.

Our ROM-based cartridge technology means you can use new AtariWriter on any ATARI® Home Computer (even 16K) for personal and business correspondence, term papers, committee reports, mailings, etc.

It also lets you choose between cassette and disk storage systems.

One very special AtariWriter feature: you can correct as you write, without switching back and forth between Create and Edit modes.

And our memory buffer offers an "undo" command to let you change your mind, and restore text you've just deleted.

Check into our remarkable AtariWriter, and our choice of letter quality and dot matrix printers, also reasonably priced, at Atari dealers. Call 800-538-8543 for dealer nearest you. In California, call 800-672-1404.

You'll do more with Atari Home Computers.

Listing 1 *(continued)*

```
21705 GOTO 21650
21710 REM ****Brick 1 x 2
21711 GOSUB 21481:GOSUB 21483:GOSU
    B 21483:GOSUB 21485:GOSUB 2148
    7:GOSUB 21487
21712 GOSUB 21502:GOTO 21502
21720 REM ***Get Length of Rectang
    le
21721 IF DF=2 THEN J=J+1:A$=SS$(J,
    1):GOTO 21723
21722 GET #1,C
21723 LN=C-48:IF LN<1 OR LN>9 THEN
     21722
21724 IF DF=1 THEN SS$(LEN(SS$)+1)
    =A$
21725 GOTO 21650
21730 REM ***Input Width for Recta
    ngle
21731 IF DF=2 THEN J=J+1:A$=SS$(J,
    1):GOTO 21733
21732 GET #1,C
21733 WD=C-48:IF WD<1 OR WD>9 THEN
     21722
21734 IF DF=1 THEN SS$(LEN(SS$)+1)
    =A$
21735 GOTO 21650
21740 REM ***Rectangle
21741 FOR I=1 TO LN:GOSUB 21481:NE
    XT I
21742 FOR I=1 TO WD:GOSUB 21483:NE
    XT I
21743 FOR I=1 TO LN:GOSUB 21485:NE
    XT I
21744 FOR I=1 TO WD:GOSUB 21487:NE
    XT I
21745 GOTO 21650
21750 REM ***Decision Box
21751 GOSUB 21488:GOSUB 21488:GOSU
    B 21482:GOSUB 21482
21752 FOR I=1 TO WD-2:GOSUB 21483:
    NEXT I
21753 GOSUB 21484:GOSUB 21484:GOSU
    B 21486:GOSUB 21486
21754 FOR I=1 TO WD-2:GOSUB 21487:
    NEXT I
21755 GOTO 21650
21760 REM ***Canned String
21761 DF=2:IF LEN(SS$)=0 THEN 2177
    0
21762 FOR J=1 TO LEN(SS$)
21763 A$=SS$(J,J)
21764 GOSUB 21450
21769 NEXT J
21770 DF=0:GOTO 21650
30000 REM ---Initialization---
30020 OPEN #1,4,0,"K:"
30030 FILE=13000
30100 REM ****Copy ROM Characters*
    **
30120 GOSUB 31300
30170 PC=START
30180 FOR I=PC TO PC+16*8-1:READ A
    :POKE I,A
30183 POSITION 0,0:? I;
30185 NEXT I
30187 POSITION 0,0:? "              ";
30190 RETURN
31080 REM ---16 Bit OR---
31090 DATA hh Lh KhLUhKT
31100 REM ---16 Bit AND---
31105 DATA hh Lh Kh%LUh%KT
31107 REM ---Block Move---
31110 DATA hh Lh Kh Nh Mh Ph O 1K
    MfKP fLfMP fNFO%OI P FP%Oh PP

31120 DATA 0,0,0,0,0,0,0,0
31130 DATA 0,0,0,0,0,0,0,255
31140 DATA 128,128,128,128,128,128
    ,128,128
31150 DATA 128,128,128,128,128,128
    ,128,255
31160 DATA 128,64,32,16,8,4,2,1
31170 DATA 128,64,32,16,8,4,2,255
31180 DATA 128,192,160,144,136,132
    ,130,129
31190 DATA 128,192,160,144,136,132
    ,130,255
31200 DATA 1,2,4,8,16,32,64,128
31210 DATA 1,2,4,8,16,32,64,255
31220 DATA 129,130,132,136,144,160
    ,192,128
31230 DATA 129,130,132,136,144,160
    ,192,255
31240 DATA 129,66,36,24,24,36,66,1
    28
31250 DATA 129,66,36,24,24,36,66,2
    55
31260 DATA 129,194,164,152,152,164
    ,194,129
31270 DATA 129,194,164,152,152,164
    ,194,255
31300 DIM M2$(18),M3$(18),M4$(54),
    A$(1),SS$(50)
31305 POKE 106,PEEK(106)-16:START=
    256*PEEK(106):POKE 82,0:GRAPHI
    CS 0
31330 READ M2$,M3$,M4$:O16=ADR(M2$
    ):A16=ADR(M3$):BMOVE=ADR(M4$)
31350 POKE START,0:U=USR(BMOVE,STA
    RT,START+1,1023):REM  ZERO CS 
    MEMORY
31360 U=USR(BMOVE,256*PEEK(756),ST
    ART,1024):REM  MOVE ATARI CS D
    OWN
31370 POKE 756,144:RETURN
```

MICRO™

ATARI PROGRAMMABLE CHARACTER EDITOR

by Tom Marshall

Type in and SAVE the program shown in listing 1. Be sure the characters in lines 32010, 32110, 32210, and 32310 are typed exactly as shown in the listing. These represent machine-language instructions, where even the slightest error is likely to result in a crash. SAVE the program before you try it, or you may end up having to type it all in again!

LOAD and RUN the program. The screen will clear and the bottom half of the screen will fill with a full character set display. Then the dot matrix representation of the space character will be drawn out on the top half of the screen, with a cursor in the upper left corner. The CTRL key, together with the arrow keys, will allow you to move the cursor to any point within the 8 × 8 matrix. The space bar *toggles* the state of the pixel represented at the cursor position: if the pixel is on, the space will turn it off; if the pixel is off, the space will turn it on. Incidentally, the corresponding bit in the RAM character generator memory will be changed, and the actual size character (and its reverse image) in the display below will change, as well.

The ESC key will cause a one-line menu to be displayed. ESC returns you to the character editing function. L and S allow you to LOAD and SAVE character sets. C allows you to specify the character code (see the table in your manual) of the character you wish to edit. *When you are LOADing or SAVEing your character set, be sure to include the appropriate prefix (D:, D1:, etc.).*

Because the display list is altered in this program, be sure to hit the SYSTEM RESET key every time the program breaks (such as on an error or when the BREAK key is hit). Otherwise, you are likely to lose control of the computer.

Program Description

Line 10 dimensions all the strings and arrays as required by the program. Line 20 calls the initialization routine (starting at line 30000) and then skips to the matrix display segment (beginning at line 500).

Lines 100-160 perform the actual character editing function. In line 100 the keyboard is checked for a key. Any key whose code doesn't fall in the range 27 to 32 is rejected, and an immediate branch back to the beginning of the line is taken. Line 105 tests for the ESC key, which causes a branch to line 1000, where the menu is offered. Line 110 tests for a non-space character (one of the four arrow keys) and branches to line 150. The lines in between handle the space key. Line 115 calculates the power of 2 appropriate for the current cursor position. The FOR...NEXT loop is necessary because of the inaccuracy in the exponentiation function. In line 120, L is calculated as the address of the current byte in the character generator RAM. START is the beginning of the character RAM and CHAR is the character number. The variable Y indicates the vertical position of the cursor as well as the proper byte within the character's definition. P is the actual contents of L. B is calculated in the machine-language AND routine. The variable A16 contains the address of the routine; P is the appropriate byte in the character definition; and Z is the power of 2 corresponding to the current cursor position. The machine-language routine performs a binary AND, determining whether the bit is on or off. Because the space key performs a toggle function, the current state of the bit must be reversed. First, the screen display is changed in lines 122-124. In line 126 the position is updated by incrementing X. If X already equals 7, then the expression X=7 evaluates to 1, and the whole statement evaluates to 0. Finally, the exclusive OR routine is called, which toggles the specified bit.

Line 150 is where the flow goes for any of the cursor movement keys. Since the codes range between 28 and 31, subtracting 28 from C results in a number in the range 0-3. This number is used as an index into the arrays DX( ) and DY( ) to get the proper adjustments to X and Y. The remainder of the line, plus lines 152-156, make sure that X and Y stay in range. Line 200 positions the cursor after the X and Y calculations are complete, and finally GOTO 100 returns back to get another key.

Lines 500-550 print the matrix pattern for the character. Each bit in each byte of the character's definition is checked, and a solid ball is PRINTed if the bit tested is on, and a period is PRINTed if the bit is off. The bits are checked from low to high and, as a result, the characters are PRINTed from right to left in each row. To accomplish this, each character must be followed by two cursor-left characters. After the whole matrix is printed, the cursor is POSITIONed at the upper left by branching to line 200. (X and Y still have the 0 values they received in line 30080.)

Lines 1000-1320 handle the user input when the ESC key has been pressed. In line 1000 the cursor is turned on first by POKEing a 1 into 752. The message is PRINTed and the cursor is turned off while awaiting the user's response, elicited by the GET# in line 1010. An ESC sends the flow back to 200. The other acceptable keys are handled in lines 1020-1040. Any others are rejected by branching back to line 1010.

The LOAD (1100), SAVE (1200), and character-input (1300) routines are straightforward. LOADing consists of reading the file byte-by-byte and POKEing them into the proper place in memory. SAVEing involves PUTing the characters to the file. Both routines use the file name input routine starting at line 1300. The character code input routine is a simple INPUT statement. A test is made in line 1310 for the

(Continued on next page)

proper range (0 to 127). Subroutine 3000 blanks out the prompt line when it is no longer needed.

The subroutine starting at line 30000 performs the initialization. In line 30000 the top of available RAM pointer is lowered by four pages by subtracting four from the contents of address 106. Next graphics 0 is invoked and the value of the variable START is set to the new contents of 106. POKEing a 0 into 82 changes the left margin from 1 to 0. DL is set to the address of the display list, contained in low/high order in 560 and 561.

In line 30010 the four strings that contain the machine-language routines are READ into the strings M$, M1$, M2$, and M3$. The addresses of these four strings are then computed using the ADR function and placed in the descriptively named variables A16, X16, DLI, and BMOVE. The high and low bytes of DLI are stored in the variables M and L. In line 30020, these values are POKEd into 512 and 513, the system's interrupt pointer. Then the ANTIC's interrupt pointer is POKEd with $C0.

In line 30030, the keyboard is OPENed as a device for the GET#1 statement in line 100. Line 30040 zeroes the character memory by calling the block-move routine. A zero is first POKEd into START. Then the whole character memory is copied up one byte in memory. Since the zero is the first to be moved, all 1024 bytes end up containing zeroes. The block move is used again in line 30042 to copy the ROM characters into the RAM area reserved at START.

The message "CHARACTER #" is printed in the proper place at the top of the screen in line 30050. In line 30060 the direction-displacement arrays DX( ) and DY( ) are filled.

The proper address for the screen origin is extracted from the display list and placed in the variable ST. Lines 30060-30076 POKE the entire 128-character set into screen memory in rows of 32, followed by the 128 reverse-field images. Finally the coordinates for the matrix display are set to zero, the character CH is set to zero, and FILE is set to the line number of the beginning of the file-name input routine (2000). The remaining lines contain DATA statements used by READ statements above.

The four machine-language routines stored in strings are disassembled in listings 2, 3, 4, and 5.

Listing 1

Atari Programmable Character Editor

```
10 DIM M$(18),M2$(18),M3$(11),M4$(
   54),DX(3),DY(3),F$(15)
20 GOSUB 30000:GOTO 500
100 GET #1,C:IF C<27 OR C>32 THEN
    100
105 IF C=27 THEN 1000
110 IF C<>32 THEN 150
115 Z=1:IF X<7 THEN FOR Q=1 TO 7-X
    :Z=Z*2:NEXT Q
120 L=START+CHAR*8+Y:P=PEEK(L):B=U
    SR(A16,P,Z)
122 POSITION X+16,Y+3:IF B THEN PR
    INT ".";:GOTO 126
124 PRINT "●";
126 X=X+1-8*(X=7):POSITION X+16,Y+
    3:PRINT "→←";
130 POKE L,USR(X16,P,Z):GOTO 100
150 X=X+DX(C-28):Y=Y+DY(C-28):IF X
    >7 THEN X=0:Y=Y+1
152 IF X=-1 THEN X=7:Y=Y-1
154 IF Y>7 THEN Y=0
156 IF Y=-1 THEN Y=7
200 POSITION X+16,Y+3:PRINT "→←";:
    GOTO 100
500 POSITION 26,1:PRINT CHAR;"   ";
502 FOR BYTE=0 TO 7:N=PEEK(START+C
    HAR*8+BYTE):POSITION 23,3+BYTE
510 FOR BIT=0 TO 7
520 QN=N:N=INT(N/2):IF (QN-N*2) TH
    EN PRINT "●←←";:GOTO 540
530 PRINT ".←←";
540 NEXT BIT
550 NEXT BYTE:POSITION 0,12:PRINT
    "↑↑↓↓";:GOTO 200
1000 POKE 752,1:POSITION 0,12:PRIN
     T "↑↓..............S, L, C, OR %%
     ...............":POKE 752,0
1010 GET #1,C:IF C=27 THEN GOSUB 3
     000:GOTO 200
1020 IF CHR$(C)="L" THEN 1100
1030 IF CHR$(C)="S" THEN 1200
1040 IF CHR$(C)="C" THEN 1300
1050 GOTO 1010
1100 REM ---LOAD---
1102 GOSUB FILE:OPEN #2,4,0,F$
1110 FOR Q=START TO START+1023:GET
      #2,A:POKE Q,A:NEXT Q
1120 CLOSE #2:CHAR=0:POKE 54286,19
     2:GOTO 500
1200 REM ---SAVE---
1202 GOSUB FILE:OPEN #2,8,0,F$
1210 FOR Q=START TO START+1023:PUT
      #2,PEEK(Q):NEXT Q
1220 CLOSE #2:CHAR=0:POKE 54286,19
     2:GOTO 200
1300 PRINT "↑↓ENTER NEW CHARACTER.
     ..";
1310 INPUT CHAR:IF CHAR>255 OR CHA
     R<0 THEN 1300
1320 GOTO 500
2000 PRINT "↑↓ENTER FILENAME...";:
     INPUT F$:RETURN
3000 POKE 752,1:POSITION 0,12:? "↑
     ↓
                        ":POKE 752,0
3010 RETURN


30000 POKE 106,PEEK(106)-4:GRAPHIC
      S 0:START=256*PEEK(106):POKE 8
      2,0
30002 DL=256*PEEK(561)+PEEK(560)
30004 POKE DL+10,130
30010 READ M$,M2$,M3$,M4$:A16=ADR(
      M$):X16=ADR(M2$):BMOVE=ADR(M4$
      )
30012 DLI=ADR(M3$):M=INT(INT(DLI)/
      256):L=DLI-256*M
30020 POKE 512,L:POKE 513,M:POKE 5
      4286,192
30030 OPEN #1,4,0,"K:"
30040 POKE START,0:U=USR(BMOVE,STA
      RT,START+1,1023):REM  ZERO CS
      MEMORY
30042 U=USR(BMOVE,256*PEEK(756),ST
      ART,1024):REM  MOVE ATARI CS D
      OWN
30050 POSITION 14,1:PRINT "CHARACT
      ER #"
30060 FOR Q=0 TO 3:READ DX,DY:DX(Q
      )=DX:DY(Q)=DY:NEXT Q
30070 ST=256*PEEK(DL+5)+PEEK(DL+4)
      :CH=0
30072 FOR Q=604 TO 859 STEP 40
30074 FOR QQ=Q TO Q+31:POKE ST+QQ,
      CH:CH=CH+1:NEXT QQ
30076 NEXT Q
30080 X=0:Y=0:CHAR=0:FILE=2000
30090 RETURN
32000 REM ---16 BIT AND---
32010 DATA hh LhKh%L Uh%K T◆
32100 REM ---16 BIT XOR---
32110 DATA hh LhKhEL UhEK T◆
32200 REM ---DLI ROUTINE---
32210 DATA H%j  T  The
32300 REM ---BLOCK MOVE ROUTINE---
32310 DATA hh LhKhNhMhPhO ♥1K
      MfKP fLfMP fNFO%OI P FP%Oh PP◆
      ◆
32.00 REM --DIRECTION DISPLACEMENT
      S--
32410 DATA 0,-1,0,1,-1,0,1,0
```

List 2

```
*       16 BIT AND
        PLA
        PLA
        STA     $CC
        PLA
        STA     $CB
        PLA
        AND     $CC
        STA     $D5
        PLA
        AND     $CB
        STA     $D4
        RTS
```

List 3

```
*       16 BIT XOR
        PLA
        PLA
        STA     $CC
        PLA
        STA     $CB
        PLA
        EOR     $CC
        STA     $D5
        PLA
        EOR     $CB
        STA     $D4
        RST
```

List 4

```
*       DLI     ROUTINE

        PHA
        LDA     $6A
        STA     $D40A
        STA     $D409
        PLA
        RTI
```

List 5

```
*       BLOCK MOVE ROUTINE

        PLA
        PLA
        STA     $CC
        PLA
        STA     $CB
        PLA
        STA     $CE
        PLA
        STA     $CD
        PLA
        STA     $D0
        PLA
        STA     $CF
        LDY     #$00
AGAIN   LDA     ($CB),Y
        STA     ($CD),Y
        INC     $CB
        BNE     THERE
        INC     $CD
THERE   INC     $CD
        BNE     SKIP
        INC     $CE
SKIP    DEC     $CF
        LDA     $CF
        CMP     #$FF
        BNE     HERE
        DEC     $D0
HERE    LDA     $CF
        ORA     $D0
        BNE     AGAIN
        RTS
```

MICRO

MODE 10 IN MODE 0

by Paul Swanson

The mode 10 painter program, published in the July through September issues of Micro magazine, afforded a simple method of creating very colorful, detailed screen displays on your Atari computer. It even allowed the screens to be stored on disk or cassette in a form easily read into other programs you may create. The biggest problem with mode 10 screens is that they consume about 8K of valuable memory, restricting severely the number of screens that can be stored in memory at any given time.

In many cases, this problem can be lessened substantially. By taking some of the redundancy out of the screens, many mode 10 screens can be reduced to about 2K of memory. This allows four times as many screens to be stored in the same area in memory.

This reduction is done using character graphics. The method used in character graphics is the same as the one used to put letters, numbers and other symbols on a normal text screen. In the screen image in memory are codes for, in the case of a text screen, each of the letters, etc., that are to be displayed. The computer looks up the image for each character in a character set and displays the pattern of dots on the screen as defined there. It uses the ASCII value of the character to calculate where the image is stored in the character set.

If graphics patterns are stored as the images in the character set instead of character images, then the resulting display will be graphics rather than text. A simple example of this can be seen using the CTRL characters on a normal text screen. Although this is the graphics equivalent of a mode 8 screen, one other slight change causes the computer to interpret the screen as a GTIA screen instead of the equivalent of a mode 8 screen.

In order to have the equivalent of a GTIA screen in character graphics, the character set and the characters in the screen image in memory must be defined. The character set, which contains 128 characters, requires 1024 bytes, or exactly 1K, of memory. The screen is the same as a mode zero screen and requires 960 bytes. This adds up to 64 bytes short of 2K. Most of those 64 bytes will be used to store the required display list, which tells the computer how to interpret the screen, where the screen is and a few other essential facts.

This method will work well on screens that are not very complex because it requires a lot of redundancy. Without the redundancy, there would not be enough characters in the character set to contain all of the required dot combinations. It will therefore not work on many mode 10 screens. However, the tremendous memory savings available make it worth the effort of attempting the conversion.

There is a method of converting more complex screens to character screens, but it involves multiple character sets and display list interrupts. That, in turn, requires more memory per screen and a good understanding of machine language and interrupts.

The Program

The conversion program in Listing 1 displays the resulting character graphics screen as it is being defined, showing the progress of the conversion. The program begins by setting the strings relative to a 1K boundary because character sets must start on a 1K boundary, display lists may not cross a 1K boundary and screens may not cross a 4K boundary.

Starting the strings on a 1K boundary allows adherence to all of these restrictions. The character set is defined first so that it starts on the 1K boundary. The next string is the screen, which starts on the next 1K boundary. This ends 64 bytes before the next 1K boundary, so the display list (which may not cross a 1K boundary) is assigned those last 64 bytes. Actually, the display list will require fewer than 64 bytes, but leaving the string pointer on a 1K boundary makes it easier to add to the program later if the functions added require some relationship to a 1K boundary.

The display list is defined next. The three lower case "p" characters are "blank 8 lines" instructions, telling the computer to leave the lines that are above the top of a normal mode zero screen blank. These can be displayed in a program, but are above the top of a normal television screen and are usually not used. The upper case "B" character tells the computer to display the first line as an IR mode two line, which is the same as a mode zero line (the "instruction register" mode numbers are not the same as the "operating system" mode numbers used by BASIC), starting at the address in memory defined by the next two characters.

These next two characters are CTRL-commas, which are ASC character zeroes. These are used as placeholders. The actual address will be calculated and inserted into the string later in the program. The place for the address is followed by 23 CTRL-B's, which tell the computer to continue the display 23 more lines, all in IR mode two. This is followed by an uppercase "A" character, which is a "jump on vertical blank" instruction, ending the display list.

A jump on vertical blank instruction stops the display information until the television finishes the current scan of the screen and returns to start the next scan. The computer will expect two more characters to follow this character telling it where to begin the information for the next scan. These two characters will refer back to the beginning of this display list to repeat the same set of instructions. This is done at lines 170 and 180. Line 190 fills in the address of the beginning of the actual screen in memory.

After all of that setting up is completed, the remaining strings and the array used in this program are DIMed at line 230. BUF$ holds the mode 10 screen as created by the painter program. The SCOL array stores the nine colors. TEST$ is used to compare patterns in the character set, FILE$ holds the file specification of the mode 10 file and LBUF$ is used to read and write files.

Lines 270 through 320 display an introductory message on the screen. Lines 360 through 380 INPUT the mode 10 file specification. Lines 390 and 400 check to see if the file is valid, OPENing it if it is. This is followed by lines 410 through 430 reading the mode 10 file into memory. The progress of the loading is indicated by dots on the screen.

When the loading is complete, the conversion begins immediately. The character set is cleared in line 480 and the screen image is cleared in line 490. Line 500 tells the computer to look at the display list in DL$ (POKE 560 and 561), interpret it as mode 10 (POKE 623,128) and use the character set starting at the address of CSET$. The colors are set by the FOR/NEXT loop at line 510.

Now the conversion can begin. One character image is predefined. That is the one corresponding to CHR$(0), which is defined as background image. Each character will be two dots (pixels) wide by eight lines. Because of the way mode 10 works, that interprets to an image of eight bytes, or characters, in the character set. The first eight bytes are left as ASC code zeroes (the heart).

Three variables are used to control the progress of conversion. ROW is the row number on the character screen. It starts at zero and ends with 23 at the bottom of the screen. CLM is the column number, which is actually half the corresponding column value of the mode 10 screen. It starts at zero and ends at 39, after which ROW is incremented and CLM is set back to zero. CNO is the number of characters defined. SC is also used to keep track of the character number being defined on the screen in memory. This could actually be calculated from ROW and CLM, but keeping this counter avoids some multiplication, resulting in a slightly faster loop.

These variables keep track of the bytes on the mode 10 screen that are in each character position. The ones corresponding to ROW and CLM are extracted from the screen by line 560 and stored in TEST$. This string is used for comparison with each character in the set that has been defined so far. If a match is found, the character on the screen is set to refer to the image found in the character set. If a match is not found, the image is added to the character set and the new character number is used in that place on the screen.

This conversion loop can end in one of two ways. If all goes well, the loop ends when the entire conversion is completed. In that case, the file is saved. If the screen requires more than 128 characters, the loop stops when 128 characters have been defined. In that case, the computer will continually buzz the keyboard speaker until a key is hit on the keyboard, or SYSTEM RESET is pressed. SYSTEM RESET will cancel the conversion. Any keyboard key will cause the computer to save the screen partially converted.

Using The Program

In trying to convert your mode 10 displays to character graphics screens, you may find the results somewhat discouraging. Many of your screens will probably not successfully convert. You may run out of characters halfway down the screen.

Keeping in mind how the screen will be converted will help substantially. Beginning at the upper left corner of the screen, each character will occupy the space of two dots horizontally by eight vertically. Placing corners and sides of objects on character boundaries reduces substantially the number of characters required in the conversion. Adding a selection to, for example, the save/load section, which merely displays the numbers of the current cursor position, will avoid much counting. The conversion method is not perfect, but can be used effectively on screens that are simple and/or placed effectively in relation to the character boundaries.

Full screens may not alway be required and half or even ¾ screens are much more likely to be converted completely. The display list of the program that will use the data formed by this program can center the screen by placing more lower case "p" characters at the beginning of the display list and the display list should have fewer CTRL-B's in proportion to the size of the screen being displayed.

Listing 2 allows selection of partial screens and assumes that the screens were all drawn starting at the top of the screen in the mode 10 painter program. Compare Listing 2 to Listing 1 to see how to implement the functions required in a program that will use the data files created by the conversion.

(Continued on next page)

Advanced Programming

A little machine language and a few alterations can allow more complex mode 10 screens to be converted to 3K character graphics screens. Alter the conversion program to convert the top half of the screen with one character set, then use a second character set and start the interpretation over for the bottom half of the screen. Save the screen and the resulting two character sets.

The program that displays the data will require a display list interrupt at line 11 on the screen, so use inverse video for the 10th CTRL-B in the display list. Set up a machine language routine in a protected area in memory (not in a BASIC string because these move if you go into immediate mode. Page 6 is much safer).

This routine should push the accumulator onto the stack and load it with the page number of the second character set. Next, it should STA to WSYNC (location $D40A) to wait until the end of the current scan line. At that point, STA $D409 will change character sets. After that, just PLA and RTI.

Put the page number of the first character set in location 756 (decimal) from the BASIC program. When you switch to the new display list and enable the display list interrupts, the computer will display the upper half of the screen using the first character set and the lower half using the second character set. Advanced programs using GTIA screens will save a lot of memory using this method. Some memory is used implementing the method, but reducing an 8K screen to 3K more than compensates for this.

A display list interrupt with a partial screen can also provide a mode 10 screen with a text window. Listing 3 uses a display list interrupt to cancel the GTIA interpretation halfway down the screen, which converts the bottom half of the screen to a mode zero screen.

A blank line instruction is used because five things are changed, including the GTIA interpretation, the character set and two colors. The problem this avoids is that it takes more time than is available in a horizontal blank period to change five things. If some of the changes happened after the scan line started, the result would create an undesired effect on the screen at that point.

The fifth item altered in the display list interrupt is that the background color is changed to blue. This is done so that the blank line will look like part of the text screen. It is done first because none of the other changes will affect the blank line. One line on the screen takes place in more than 120 clock cycles, which allows much more than this interrupt routine requires.

To avoid converting the characters for the lower part of the screen from ATASCII to the codes required for the screen, the upper half of the BASIC mode zero screen is used to form the text part of the screen. That way, all text can be put on the screen using PRINT from BASIC.

Use Of The Program

Converting mode 10 screens to character graphics screens and gaining 6K back from an 8K screen opens up many new uses for the painter program. On a 48K Atari computer, in BASIC, eight of these screens would not be a great burden. Writing only two bytes to select a display list selects these screens very rapidly. This opens the door to some very colorful, easy to create static graphics programs, like presentations. It also provides the speed required for short, simple animation routines. Using half screens instead of full screens for the animation allows 1.5K per screen (the character set will still require 1K). Eleven half screens, with proper advanced planning, can be stored in the 16K suggested for the eight 2K screens. Use some color indirection and the number of screens multiplies, although you are trading some of the colors for the animation.

Listing 1

```
10 REM ***************************
20 REM * GTIA Conversion Program *
30 REM * to   convert   mode  10 *
40 REM * screens  to   character *
50 REM * graphics screens.       *
60 REM *  by Paul S. Swanson     *
70 REM ***************************
80 REM
90 REM ** SET BOUNDARY & DIMS **
100 REM *
110 DIM X$(1):A=ADR(X$):B=INT(A/1024+1)
*1024:DIM F$(B-A-1)
120 DIM CSET$(1024),SC$(960),DL$(64)
130 REM
140 REM ** DEFINE DISPLAY LIST **
150 REM
160 DL$="pppB♥♥|||||||||||||||||||||||A
"
170 A=ADR(DL$):DLHI=INT(A/256):DLLO=A-D
LHI*256
180 DL$(LEN(DL$)+1)=CHR$(DLLO):DL$(LEN(
DL$)+1)=CHR$(DLHI)
190 A=ADR(SC$):SCHI=INT(A/256):SCLO=A-S
CHI*256:DL$(5,5)=CHR$(SCLO):DL$(6,6)=CH
R$(SCHI)
200 REM
210 REM ** REMAINING DIM'S **
220 REM
230 DIM BUF$(8000),SCOL(8),TEST$(8),FIL
E$(14),LBUF$(80)
240 REM
250 REM ** SET UP SCREEN **
260 REM
270 GRAPHICS 0:? "          ** CONVERSION P
ROGRAM **":?
280 ? "This program converts data files
":? "created by the Mode 10 painter"
290 ? "program to character graphics sc
reens":? "and stores the character set
and"
300 ? "new screen on disk or cassette."
:?
310 ? "Program written by Paul S. Swans
on"
320 ? "for publication in Micro magazin
e":?
330 REM
340 REM ** GET FILESPEC & LOAD FILE **
350 REM
360 ? "Enter filespec of the Mode 10"
370 ? "file you wish to convert:"
380 INPUT FILE$:TRAP 400
390 OPEN #3,4,0,FILE$:GOTO 410
400 ? "ERROR - INVALID FILESPEC":? "↑↑
";:GOTO 380
```

Listing 1 *(continued)*

```
410 TRAP 20000:GRAPHICS 0:? "         ** C
ONVERSION PROGRAM **":? "Now loading fi
le";
420 FOR I=0 TO 8:INPUT #3,SCOL:SCOL(I)=
SCOL:NEXT I:? ".";
430 FOR I=1 TO 7680 STEP 80:INPUT #3,LB
UF$:BUF$(I,I+79)=LBUF$:? ".";:NEXT I:?
:CLOSE #3:TRAP 40000
440 REM
450 REM ** BEGIN CONVERSION **
460 REM ** DISPLAY NEW SCREEN **
470 REM
480 CSET$="♥":CSET$(1024)="♥":CSET$(2)=
CSET$
490 SC$="♥":SC$(960)="♥":SC$(2)=SC$
500 POKE 560,DLLO:POKE 561,DLHI:POKE 62
3,128:POKE 756,B/256
510 FOR I=0 TO 8:POKE 704+I,SCOL(I):NEX
T I:CNO=1
520 ROW=0:CLM=0:SC=1
530 REM
540 REM ** GET CHARACTER FORMAT **
550 REM
560 FOR I=0 TO 7:TEST$(I+1,I+1)=BUF$(CL
M+(ROW*8+I)*40+1):NEXT I
570 REM
580 REM ** CHECK IF ALREADY EXISTS **
590 REM
600 FOR TEST=0 TO CNO-1:CSTART=TEST*8+1
:IF TEST$=CSET$(CSTART,CSTART+7) THEN P
OP :GOTO 630
610 NEXT TEST:REM --NOT FOUND--
620 CSTART=CNO*8+1:CSET$(CSTART,CSTART+
7)=TEST$:TEST=CNO:CNO=CNO+1:IF CNO>127
THEN 700
630 SC$(SC)=CHR$(TEST):SC=SC+1:CLM=CLM+
1:IF CLM>39 THEN CLM=0:ROW=ROW+1
640 IF ROW<24 THEN 560
650 FILE$(LEN(FILE$)+1)=".CVT":OPEN #3,
8,0,FILE$
660 FOR I=0 TO 8:? #3;SCOL(I):NEXT I
670 FOR I=1 TO 960 STEP 40:? #3;SC$(I,I
+39):NEXT I:FOR I=1 TO 1024 STEP 64:? #
3;CSET$(I,I+63):NEXT I
680 GRAPHICS 0:? "          ** CONVERSION
PROGRAM **":?
690 ? :? :? :? "Screen converted.":? "T
otal: ";CNO;"characters.":STOP
700 ? "←";:IF PEEK(764)=255 THEN 700
710 GOTO 650
20000 ? :? "Error in loading file.":STO
P
```

Listing 2

```
10 REM ****************************
20 REM * Partial screen display *
30 REM * for mode 10  converted *
40 REM * data files.            *
50 REM ****************************
60 REM
70 REM ** SET BOUNDARY & DIMS **
80 REM
90 DIM X$(1):A=ADR(X$):B=INT(A/1024+1)*
1024:DIM F$(B-A-1)
100 DIM CSET$(1024),SC$(960),DL$(64),FI
LE$(14),LBUF$(64),SCOL(8)
110 REM
120 REM ** INPUT PARAMETERS & FILE **
130 REM
140 GRAPHICS 0:? "    ** PARTIAL SCREEN
EXAMPLE **";?
150 ? "Enter number of lines to display
":INPUT DISPLAY
160 ? "Enter number of lines to skip":I
NPUT SKIP
170 ? "Enter file specification":INPUT
FILE$
180 CLOSE #3:OPEN #3,4,0,FILE$
190 FOR I=0 TO 8:INPUT #3,SCOL:SCOL(I)=
SCOL:NEXT I
200 IF SKIP>0 THEN FOR I=1 TO SKIP:INPU
T #3,LBUF$:NEXT I
210 FOR I=1 TO 960-SKIP*40 STEP 40:INPU
T #3,LBUF$:SC$(I,I+39)=LBUF$:NEXT I
220 FOR I=1 TO 1024 STEP 64:INPUT #3,LB
UF$:CSET$(I,I+63)=LBUF$:NEXT I
230 CLOSE #3
240 REM
250 REM ** FORM DISPLAY LIST **
260 REM
270 A=ADR(SC$):SCHI=INT(A/256):SCLO=A-S
CHI*256
280 DL$="ppp":IF SKIP>0 THEN FOR I=1 TO
 SKIP:DL$(LEN(DL$)+1)="p":NEXT I
290 DL$(LEN(DL$)+1)="B":DL$(LEN(DL$)+1)
=CHR$(SCLO):DL$(LEN(DL$)+1)=CHR$(SCHI)
300 FOR I=1 TO DISPLAY-1:DL$(LEN(DL$)+1
)=" |":NEXT I:DL$(LEN(DL$))="A"
310 A=ADR(DL$):DLHI=INT(A/256):DLLO=A-D
LHI*256:DL$(LEN(DL$)+1)=CHR$(DLLO):DL$(
LEN(DL$)+1)=CHR$(DLHI)
320 REM
330 REM ** DISPLAY SCREEN **
340 REM
350 POKE 560,DLLO:POKE 561,DLHI:REM --D
ISPLAY LIST POINTER
360 POKE 756,B/256:REM --CHARACTER SET
370 FOR I=0 TO 8:POKE 704+I,SCOL(I):NEX
T I:REM --POKE COLORS
380 POKE 623,128:REM --ENABLE GTIA
390 GOTO 390
```

Listing 3

```
10 REM ******************************
20 REM * Mode 10 conversion using *
30 REM * a display list interrupt *
40 REM * to enable a text window. *
50 REM ******************************
60 REM
70 REM ** SET BOUNDARY & DIMS **
80 REM
90 DIM X$(1):A=ADR(X$):B=INT(A/1024+1)*
1024:DIM F$(B-A-1)
100 DIM CSET$(1024),SC$(480),DL$(64),LB
UF$(64),FILE$(14),SCOL(8)
110 REM
120 REM ** LOAD FILE **
130 REM
140 CLOSE #3:OPEN #3,4,0,"D:BLOCK.CVT":
REM ---INSERT NAME OF FILE IN THE OPEN
STATEMENT ---
150 FOR I=0 TO 8:INPUT #3,SCOL:SCOL(I)=
SCOL:NEXT I
160 FOR I=1 TO 480 STEP 40:INPUT #3,LBU
F$:SC$(I,I+39)=LBUF$:NEXT I
170 FOR I=481 TO 960 STEP 40:INPUT #3,L
BUF$:NEXT I
180 FOR I=1 TO 1024 STEP 64:INPUT #3,LB
UF$:CSET$(I,I+63)=LBUF$:NEXT I
190 REM
200 REM ** SET UP DISPLAY LIST **
210 REM
220 DL$="pppB♥♥|||||||||█♥B♥♥|||||||||
 ||A"
230 DL=PEEK(560)+PEEK(561)*256:DL$(20,2
0)=CHR$(PEEK(DL+4)):DL$(21,21)=CHR$(PEE
K(DL+5))
240 A=ADR(SC$):SCHI=INT(A/256):SCLO=A-S
CHI*256:DL$(5,5)=CHR$(SCLO):DL$(6,6)=CH
R$(SCHI)
250 A=ADR(DL$):DLHI=INT(A/256):DLLO=A-D
LHI*256:DL$(LEN(DL$)+1)=CHR$(DLLO):DL$(
LEN(DL$)+1)=CHR$(DLHI)
260 REM
270 REM ** PUT DLI IN PLACE **
280 REM
290 RESTORE :LOC=1536
300 READ N:IF N<256 THEN POKE LOC,N:LOC
=LOC+1:GOTO 300
310 REM
320 REM ** SET UP TEXT SCREEN **
330 REM
340 GRAPHICS 0:? :? "         ** MODE 10
SCREEN **";?
350 ? "Using a display list interrupt t
o":? "provide a text window.";
360 REM
370 REM ** ENABLE SCREEN **
380 REM
390 POKE 560,DLLO:POKE 561,DLHI:REM --D
ISPLAY LIST POINTER
400 POKE 512,0:POKE 513,6:REM --DLI VEC
TOR
410 FOR I=0 TO 8:POKE 704+I,SCOL(I):NEX
T I:REM --COLORS
420 POKE 623,128:REM --GTIA INTERPRETAT
ION
430 POKE 54286,192:REM --ENABLE DLI
440 POKE 756,B/256:REM --CHARACTER SET
450 GOTO 450
1000 REM
1010 REM ** DLI ROUTINE **
1020 REM -----ASSEMBLER LISTING-----
1030 REM *48          PHA
1040 REM *A9 A4       LDA #$A4
1050 REM *8D 0A D4    STA $D40A WSYNC
1060 REM *8D 1A D0    STA $D01A COL4
1070 REM *8D 18 D0    STA $D018 COL2
1080 REM *A9 0A       LDA #$0A
1090 REM *8D 17 D0    STA $D017 COL1
1100 REM *A9 E0       LDA $E0
1110 REM *8D 09 D4    STA $D409 CHBASE
1120 REM *A9 00       LDA #$00
1130 REM *8D 1B D0    STA $D01B PRIOR
1140 REM *68          PLA
1150 REM *40          RTI
1160 REM ---CONVERTED TO DECIMAL---
1170 DATA 72,169,164,141,10,212,141,26,
208,141,24,208,169,10,141,23,208,169,22
4,141,9,212
1180 DATA 169,0,141,27,208,104,64,256
```

MICRO

Atari Draw Package

by Robert M. Tripp

By typing " (shift 2), you can enter the DRAW routines — lines 21400-21770 in the screen editor listing. The double quote is a toggle, pressing it once enters the line editor, pressing it from within the DRAW routine exits to the normal screen editor. Within this routine is the capability to construct complicated line drawings with text and graphic characters included.

CTRL-2 toggles the "canned string" routine. This is checked for in lines 21412 and 21414. Lines 21451-21466 decode the direction plot and move routines as described in the general graphics section. When drawing lines across existing lines, the current line must be "ORed" with the line on the screen to form the combination of two or more lines. Since Atari BASIC doesn't have a bit by bit "OR" such as found in Commodore BASIC, it is necessary to use machine-language to do the OR routine. A machine-language AND routine is used to erase only the current line from the screen. These routines are found listed as strings in 31080-31105. The code for the AND is listed under the Atari Character Editor program. The OR merely substitutes an OR in the two lines containing ANDs. The string in line 31110 contains the codes for a block move routine. This is also listed in the character editor section.

You can use these routines in your own programming by maintaining the correct syntax when calling the subroutines. The AND, OR, and XOR routines are called in the same manner:

VAR1 = USR(ADD,VAR2,VAR3)

where ADD is the address of the string containing the routine, VAR2 and VAR3 are the variables to be ANDed, ORed or XORed, and VAR1 is the resultant of the operation. The block move is very useful as you can move memory to anywhere else in RAM. The correct syntax for this routine is:

U = USR(ADD,DEST,START,NUM)

where U is a dummy variable, ADD is the address of the string containing the block move, DEST is the address where the move will place the new information, START is the address where the memory to be moved starts, and NUM is the number of bytes to be moved. NOTE: This is a low to high move, so special effects can be obtained by overlapping the memory areas utilized. Line 31350 moves a zero into 1024 bytes of memory.

The Atari is capable of displaying 128 different characters at a time. It can also display the same 128 characters reversed. Since the line editor uses non-standard characters for displaying the different combinations of lines, some of the normal characters have to be replaced by the new definitions. Line 30180 redefines screen characters 0-15, ASCII numbers 32-47 to be these new characters. This means that those characters (and their inverse) can no longer be printed on the screen.

MICRO

MICRO™

From Here To Atari

Paul S. Swanson

This marks the beginning of my second year of writing this column for MICRO Magazine. Response has been good so far, but I am always looking for reader feedback. Mail is the only way to assure that this column continues to reflect its readers' interests.

XBASIC and SAM

XBASIC is a series of utilities for BASIC programming. Most of the complaints I have heard concerning Atari BASIC stem from the fact that it doesn't support string arrays. XBASIC not only allows you to use string arrays, but it also allows you to use the space reserved by a string as an integer array. These are only two of the many programming aids offered by this utility package.

The code that implements these functions is booted from disk and resides in memory all of the time the BASIC program is being written and while it is running. A special SAVE command puts this code, as well as the BASIC program, on disk for you. There are a few simple lines of BASIC code, supplied on the disk with the utilities, that are used to enable XBASIC.

Included among the utilities offered by XBASIC are enhanced I/O functions (GET and PUT characters, screen save/load, move screen, move memory, fill memory, double peek, etc.), timer functions, string functions (arrays, searches, get and put, integer array in string), enhanced SOUND controls, disk-directory listing without leaving BASIC, player-missile movements, and additional graphics modes. It also adds seven error messages to the list of BASIC errors to help in the use of the utilities.

XBASIC may be just the thing for the many BASIC people who have a fear of machine language. It is a useful addition to Atari BASIC and will handle a lot of the functions that I normally attribute to machine-language subroutines. In fact, it is a series of machine-language subroutines, all accessed with various USR(calls. XBASIC is available from SUPERware, 2028 Kingshouse Rd., Silver Spring, MD 20904. It is supplied with a user manual written in a reference-type format and resides on a disk along with BASIC programs that contain examples of using the utilities.

SAM (Software Automatic Mouth) is a very different utility. Also available on other computers including Apple and, in the near future, the Commodore 64, SAM sets up the computer to speak. No additional hardware is required for the Atari version. The voice is generated using the normal SOUND channels and is produced by the television speaker. As speech synthesizers go, SAM's voice is not bad. I would rate it as one of the more intelligible ones; however, all speech synthesizers tend to sound Scottish to me.

SAM can be used in a number of ways and comes with some supportive BASIC routines on the disk. One of the routines has SAM recite stored text, which includes a passage from Shakespeare and one from the Gettysburg address. Another routine allows you to type in a phrase (in normal English text,) which is converted to the phonemes and recited by SAM. Also, you can control SAM directly (using phonemes) by implementing a USR call from BASIC, so you can add SAM's voice to your own programs. SAM is also disk-based and is available from Don't Ask Software, 2265 Westwood Blvd, Suite B-150, Los Angeles, CA 90064.

Display List Interrupts

These small machine-language routines can be written and implemented in a BASIC program very easily. The example does only one, very simple task, but the display-list interrupt can be used to alter any of the hardware registers at any point in the display.

The sample program sets up a display/list interrupt that changes the color of the screen to the blue used for the normal text background. The program goes on to alter the color in the screen background color register to black. The result is a screen that is black with a blue rectangle at the bottom. Text will print in both areas normally, unaffected by the change in color.

The display list interrupt is stored in page 6, which starts at location 1536. Converted to decimal, the interrupt routine is contained in line 90. It begins with a PHA (PusH A onto the stack), decimal 72, so that it can use the register without altering it as far as the interrupted program is concerned. Next, the accumulator is loaded with 148, which is the value for the color of the normal blue background. LDA, immediate, is a 2-byte command, which is 169 in decimal. The next command, a STA, which is a three byte command, is a little more involved than just storing the contents of the A register. It stores to a special hardware register called WSYNC, which is wired to stop the processing until the current scan line is completed. Because of that, the alterations take place while the beam forming the picture is turned off.

Once the preliminary work is complete, the STA (141,24,208) puts the contents of the A register in the hardware register corresponding to SETCOLOR 2,....Then a PLA (decimal 104) restores the A register with the contents it had when the interrupt was called and the 64 is the return from the interrupt (RTI).

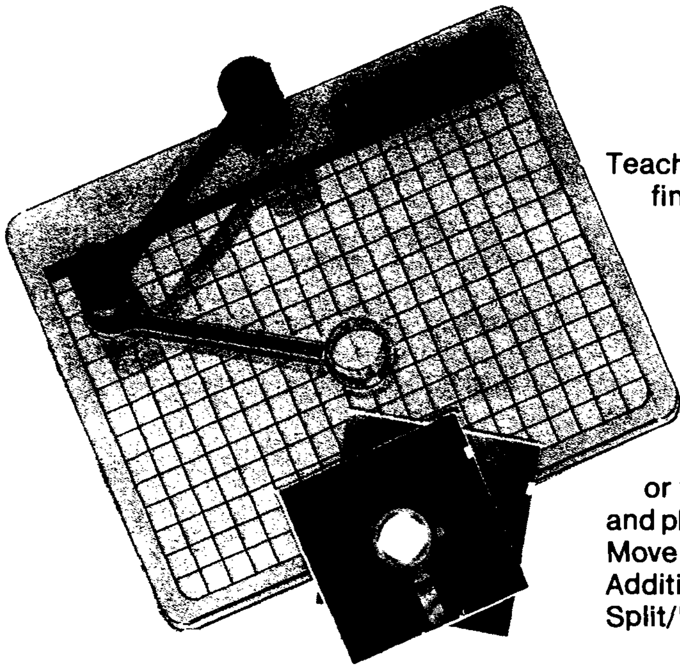
Now that the interrupt is in place, you must implement it, which requires three steps. The first step is to put the address of the routine in the display-list interrupt vector. Line 40 places 1536, which is $0600, into the proper locations. The next step defines where on the screen the interrupt will take place. This requires setting one bit in the display list. The location of the display list is at locations 560 and 561 and is read into the variable DL in line 50. Line 60 sets the highest order bit on the 12th screen line. The interrupt will start at the beginning of the last scan line of that line of characters on the screen.

Before the last step, which enables the display-list interrupt, the sample program changes the text background to black. The POKE following that writes to the non-maskable interrupt-enable register. The two highest order bits (decimal values 128 and 64) enable the display-list interrupt (value 128) and the vertical-blank interrupt (value

(Continued on page 78)

VersaWriter

FOR COMPLETE GRAHICS



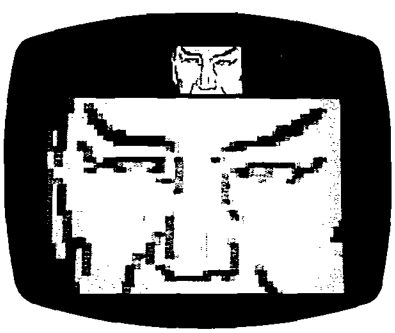
GRAPHICS TABLET AND SOFTWARE

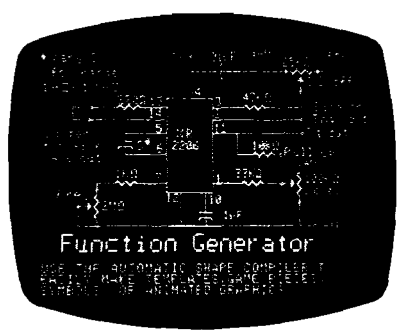
Teachers, artists, engineers, programmers & hobbiests find **VersaWriter** an easy to use tool for creating micro computer graphics. No programming experience is required. Pictures can be made by simply tracing. Even children can explore the exciting world of computer graphics. The **VersaWriter** is as limitless as your imagination.

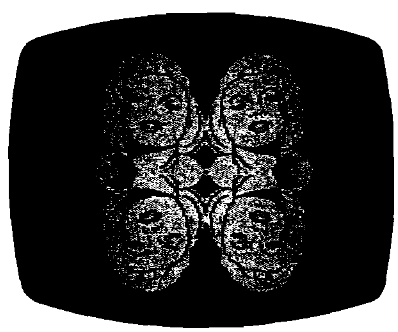
VersaWriter contains complete software for drawing with color, brushes & dots. Add text or fill in over 100 colors. Create your own shapes and place anywhere on the screen. Use Area/Distance, Move Picture, Electronic Drawing & Skeleton programs. Additional features include Reverse Picture, Scaling, Split/Full Screen, Save, Load, Erase plus much more..

Complete Hardware/Software System for Apple II/II+/IIe **$299.00**
(Also available for Atari and IBM PC)

VERSAWRITER DESIGNER PACS*







EXPANSION PAC I - Create real time shape tables in full color. Contains 350 predefined shapes for architecture, games, circuit boards, etc. Edit or define pictures using the true color HIRES microscope with up to 4x magnification. **$39.95**

*Requires VersaWriter

EXPANION PAC II - Adds 10 powerful graphics commands to **VersaWriter** including: Shrink, Magnify, Fast Picture Loader, Combine Two Screens, Color Remover, and Mirror & Upside Down Functions. Mix functions to create unusual effects. **$39.95**

EXPANSION PAC III - NEW! To be released this Fall - Animation Tool Kit. **$49.95**

VERSA COMPUTING, INC. 3541 Old Conejo Rd. #104 • Newbury Park, CA 91320 • (805) 498-1956

64). At that point in the program, the color changes will occur and the computer will be in the immediate mode. You can then enter from the keyboard other SETCOLOR 2,.... commands. The following immediate-mode statement produces an interesting effect:

```
FOR I=0 TO 15:FOR J=0 TO 15 STEP .1:SETCOLOR 2,I,J:NEXT J:NEXT I
```

To restore the screen to the normal colors without the interrupt, just press SYSTEM RESET. More can be added to the display-list interrupt, and this one will be used in later columns to show how various items are added.

Telecommunications

Using modems on personal computers is becoming popular. There are many bulletin board services across the country that allow access without charge as well as several that have rather low fees. On most of the free boards, if you have a modem set up on your computer and a terminal program running, you can access a message base, download programs that will run on your computer, play games, buy and sell used or new computer equipment and software, and even check out the local weather forecast. The message bases discuss almost anything that the callers have found of interest in the public messages and also support, through private messages or in a separate "E-Mail" section, messages to particular callers. Other than discussions of interesting topics, the message bases are also often used to request help on different computers.

These boards are often privately owned, operating out of someones home, although some are owned and operated by computer stores. Almost all of them operate at 300 baud, which is the cheapest modem on the market; many will also operate at 1200 baud, a rate that is becoming more and more popular.

If you do add telecommunications capability, I would like to hear from you. I operate a "side board" on the Outpost at 617-259-0181. It is called Atari World and supports all of the features outlined above.

Telecommunications is a very interesting function that can be added to your home computer. There is a wealth of public domain software, most of it in the games and utilities categories, that can be downloaded from the boards. The message bases, beyond providing entertainment by sporting the discussions (don't hesitate to jump in the middle of one — you will almost always be welcomed either directly or by someone countering your point of view), also provide a place for you to ask for help with your personal computer. Just leave a message stating what computer model you have (and what peripherals are on it) and explain what the problem is. Chances are you will have a response within a day or two from someone who knows how to solve your problem. If you leave it on Atari World, I'll probably answer it for you.

```
10 RESTORE :LOC=1536
20 READ N
30 IF N>256 THEN POKE LOC,N:LOC=LOC+1:GOTO 20
40 POKE 512,0:POKE 513,6
50 DL=PEEK(560)+PEEK(561)*256
60 POKE DL+16,PEEK(DL+16)+128
70 DATA 72,169,148,141,10,212,141,24,208,104,64,256
80 SETCOLOR 2,0,0:POKE 54286,192
```

MICRO

MICRO™ Atari Reviews

Product Name: **MMG BASIC Debugger**
Equip. req'd: Atari 400/800/1200XL, 16K RAM
Price: $34.95
Manufacturer: MMG Micro Software
P.O. Box 131
Marlboro, NJ 07746

Description: The *MMG BASIC Debugger* is a utility program for the Atari BASIC programmer. There are four primary components: full and split-screen editing, trace, cross-reference, and phrase search. Debugger is loaded first followed by your own Atari BASIC program. Full-screen editing is enhanced by line scrolling in both directions. Two separate screens are maintained, either or both capable of displaying independent listings, notes, or other text. Split-screen mode allows viewing, scrolling and editing two different portions of the same program on one screen. Trace functions allow for single-stepping, trace while, conditional trace, and many other options. The cross-reference utility provides a valuable listing of variables and the lines where each is used. Search will help you find any combination of characters within a program listing. This program is extremely useful and well worth the price.

Pluses: The *BASIC Debugger* is packed with features, most of which cannot be found in any other software; and it's all machine language so it requires a minimum of RAM area.

Minuses: Page 6 is free for your use, but Page 0 locations 203-207 are used by Debugger (undocumented). A system crash may result if your program uses these memory locations.

Documentation: The documentation is good. A demo program is included so that you may become familiar with all the features.

Skill level required: Advanced beginner or intermediate programmer.

Reviewer: Tim Kilby

Product Name: **Mickey in the Great Outdoors**
Equip. req'd: Atari, one joystick, 16K for cassette users, and 32K for disk users
Price: $44.95 (suggested retail)
Manufacturer: Walt Disney Educational Media Co.
500 South Buena Vista St.
Burbank, CA 91521

Description: This Walt Disney exercise for children is done in cartoon form with child-controlled semi-animation. There is an educational package available for schools and other learning systems for the same price. The educational package, in addition to the instruction and activity booklet, includes a back-up copy of the program, a teacher's guide, and a poster and five comic books dealing with computer literacy.

Pluses: Both English and Math are intrinsically covered as Mickey Mouse visually solves the problems with the help of your child. Word choices, word scrambles, and beginning numerical equations are the topics of the four exercises. Points are tabulated and the number of correct moves made by your child is monitored, so the entire experience takes on the form of a game. All of Mickey's movements are done through a joystick, therefore limited eye-hand coordination is re-enforced.

Minuses: None noted.

Documentation: The documentation is excellent. However, the rules for the exercises are meant for the parent, not the child. It is unlikely that the child will operate the program alone the first time.

Skill level required: The game is recommended for ages 7 to 10.

Reviewer: Thomas G. Marshall

Product Name: **Career Counselor**
Equip. req'd: Atari 400/800/1200XL w/32K RAM
Price: $59.95
Manufacturer: MMG Micro Software
P.O. Box 131
Marlboro, NJ 07746

Description: *Career Counselor,* as the name implies, matches general career goals with specific occupational recommendations. The user responds to questions about interests, aptitudes, salary goals, work environment, educational requirements, etc. The computer then searches its inventory of 337 occupations for compatible careers. Brief descriptions of those careers can then be printed on the screen or printer. This program should be used only as a starting point for investigating career choices, but it is fun to canvass occupational possibilities.

Pluses: The program is easy to use. It would be great for school libraries, counseling centers, job centers, or even shopping malls. Its objectivity in career choices should be welcomed.

Minuses: Three hundred and thirty-seven careers is a lot, but the career best suited to your needs may not be included. The program does not allow for the slightest error in spelling or career title.

Documentation: The documentation is good but not necessarily needed. Most users will follow the prompts and answer the questions with ease.

Skill level required: Beginner with no programming skills needed.

Reviewer: Tim Kilby

Atari Reviews *(cont.)*

Product Name: **Ultra Disassembler**
Equip. req'd: Atari 400/800/1200XL, 32K RAM
Price: $49.95
Manufacturer: Adventure International
Box 3435
Longwood, FL 32750
(305) 862-6917

Description: *Ultra Disassembler* is a labelling disassembler for the assembly-language programmer. Object code can be disassembled from portions of memory, binary disk load files, or specified disk sectors. Output can be directed to screen, disk, or printer. Source files are written with appropriate labels to make reading the code easier. References to Atari operating system locations (pages 0, 2, 3, $D0, $D2, and $E4) are labeled with the standardized names such as AUDCTL, DOSVEC, and ICCOM. The output files(s) can then be reassembled or modified and assembled with most popular assemblers. A customizer program is included for specifying pseudo-op directives for individual assemblers. For a large input file, several output files are written for linking by an assembler.

Pluses: The disassembly is fast and efficient. The source code output is as readable as disassembled code can get. But since original comments are not and cannot be included, and labels are non-descriptive (L0001, etc.), deciphering the code can be difficult.

Minuses: Not exclusive to *Ultra Disassembler*, all disassemblers have a problem dealing with raw data found in pseudo-op operands. Such data will often be interpreted as instructions. The resulting source code can be assembled but will not make sense when read.

Documentation: Documentation covers the use, limitations, and design of the disassembler.

Skill level required: Assembly-language programmer.

Reviewer: Tim Kilby

MICRO

MICRO™ Atari Software Catalog

Artworx Releases Two New Programs

GWENDOLYN, for the *Atari*, is a new graphics adventure by Marc Russell Benioff. Featuring extensive graphics (both sides of the disk) and sound, GWENDOLYN takes you into a mysterious maze of tunnels in your quest for your lost princess — GWENDOLYN.

MONKEYMATH, for the *Atari*, *Apple*, *VIC-20*, or *Commodore 64*, is that rare educational program that effectively combines an arcade game with a true learning program. Children from ages four to nine try to score the most bananas by causing a monkey to smash the correct answer as the problem moves by on a monkey-powered treadmill.

Price for **GWENDOLYN** on disk is **$27.95**. **MONKEYMATH** is **$19.95** on cassette and **$23.95** on disk. For more information contact Artworx Software Co., Inc., 150 North Main St., Fairport, NY 14450; (716)425-2883.

Einstein's MemoryTrainer Combines Wisdom of Ages with Computer Technology

The Einstein Corporation has introduced **MemoryTrainer**, an innovative software package for the *Apple*, *Atari*, and *C64*. It combines age-old wisdom with computer technology to enable users to expand their power of memory — and to have fun while doing it. Tailored to practical, career-oriented memory demands, **MemoryTrainer** enables users to improve their ability to remember faces, dates, telephone numbers, lists, and quotations, and to employ association as a memory tool.

The Einstein **MemoryTrainer** is a full-structured tutorial system with color graphics. The software package includes a comprehensive User Guide and three separate disks containing instructional materials.

The suggested retail price of the Einstein **MemoryTrainer** for the Apple computer is **$89.95**. For further information contact The Einstein Corporation, 11340 W. Olympic Blvd., Los Angeles, CA 90064; (213)477-6733.

Micro Learn Products Currently Available at Retail Stores

Micro Lab offers *English SAT 1*. This program prepares students for the verbal section of the SAT. It gives complete explanations of all answer choices, includes analogies, antonyms, sentence completion, and grammar.

For more information see your local personal computer dealer.

MICRO™

MICRO™ Atari Hardware Catalog

The Interfast-I Buffered Programmable Printer Interface

for the Atari

Advanced Interface Devices, Inc. announces the **INTERFAST-I**, a 4K buffered programmable printer interface for Atari computers. It is hardware- and software-compatible with the Atari 850 Interface Module printer port and requires no modification to the computer. The buffering reduces the time the computer must spend printing a document. The data transfer rate from the computer to the INTERFAST-I is comparable to disk transfer rates. Many program listings and text files can be dumped within a few seconds; then the computer is free to continue working while the INTERFAST-I handles the printing chores.

The INTERFAST-I can also be programmed by the user for advanced printing applications such as graphics on printers with graphics capabilities. Translation of the ATASCII EOL character to ASCII Carriage Return is also user-selectable.

The user's manual includes a discussion of programming with examples and utility programs. These and additional programs are available on an optional disk.

Advanced Interface Devices, Inc.
P.O. Box 2188
Melbourne, FL 32902

Ladco Announces the Model 250B — 250 Voltamperes with Battery Included

Ladco Development Co., Inc. announces the Model 250B, which has a rating of 250-volt amperes at 115-volts AC. The system comes complete with a 10 ampere-hour, sealed, maintenance — free battery plus the usual RAMLOK features of RFI line filtering, voltage surge suppression, over and under line voltage protection, battery charger with overcharge and short circuit protection, battery condition meter, failure indication both visual and audible, and a frequently-controlled, square-wave output, 60-cycle inverter. The inverter is contained in a separate enclosure making it possible to store in a location remote from the computer equipment.

This 250 watt RAMLOK model has sufficient capacity to power even the 16-bit computer equipments with hard disks now appearing on the market and has ample capacity for the older 8-bit models. Devices such as video terminals and point-of-sale terminals that consume less power can share the protection provided by RAMLOK.

Ladco Development Co.
P.O. Box 464,
Olean, NY 14760
(716)372-0168

MICRO™

Outsmart your computer.

Show your computer who's boss. Earn its respect. With a Datamost book.

No matter what age or ability level you're at, we have the right book that talks just to you. And your computer. Whether you own an Apple.* An Atari.* Or just about any brand.

All our books are incredibly easy to understand.

Which will make it incredibly easy to understand your computer.

What If You're Scared Of Books About Computers?

Don't be.

Our books are written in friendly, familiar American English. Highlighted with cartoons. And illustrations. So they're fun to read. As well as educational.

And there's over 30 books to choose from. Basic computer learning books to programming books to coloring books.

You won't be bombarded with complicated programming routines until you're ready for complicated programming routines. And no funny technical talk until you've reached level where you don't think it's funny.

We'll tell you what you need to know. And then, when that's understood, you can take the next step toward outsmarting your computer.

Before You Spend Big Bucks On A Computer, Spend Little Bucks.

Once you've decided, kind of, almost, nearly, what type of computer you think you'd like to buy, buy a Datamost book.

It'll help you understand the Atari or the Commodore* or the Apple of your eye.

Before you spend a lot of money. Before you take your computer home.

You'll make a better purchase decision. Because you'll understand what you're doing. And what you and your computer can do together.

So there won't be any misunderstandings to ruin your new relationship.

How To Get The Most Out Of Your Computer.

Get the most out of our minds.

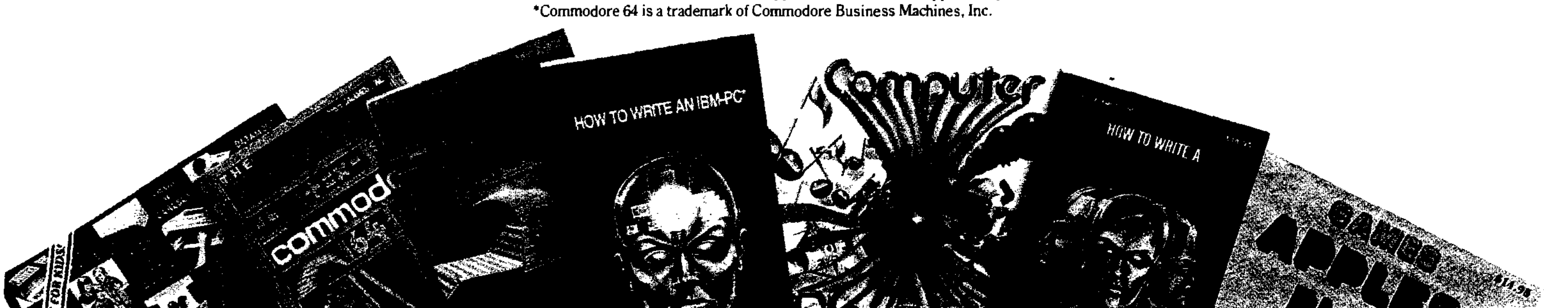
Datamost books for every computer.

Everybody. Every level.

Pretty smart, huh?

DATAMOST™ INC
The most out of our minds.™

Datamost, Inc., 8943 Fullbright Ave., Chatsworth, CA 91311, (213) 709-1202
*Atari is a trademark of Atari Computer. *Apple is a trademark of Apple Computer.
*Commodore 64 is a trademark of Commodore Business Machines, Inc.



COMPUTER MAIL ORDER

FRANKLIN

CALL
ACE 1000 Color Computer
ACE 1100 Drive & Cover for ACE 1000
ACE 1200 Computer with Disk Drive
ACE PRO PACK:
ACE 1000, Disk Drive,
80 Column Card, ACECalc &
ACEWriter II

MICRO-SCI
Apple & Franklin

A2 ....$219.00
A40 ....$299.00
A70 ....$319.00
C2 Controller $79.00
C47 Controller $89.00

RANA
Elite I (Apple/Franklin) $279.00
Elite II (Apple/Franklin) $319.00
Elite III (Apple/Franklin) $569.00

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 (Apple, IBM, CBM) $179.00
Visicalc 4 (IBM) $209.00
Visicalc Advanced IIe $309.00
Stretch Calc ....$ 89.00
Visicorp prices for IBM may vary slightly.

LJK
Letter Perfect Apple $109.00
Data Perfect Apple $75.00

AXLON
Apple/Franklin 128K Ram $299.00
Apple/Franklin Ram Disk $729.00

MPC
Bubdisk (128K Non Volitare) $649.00

JOYSTICKS
WICO

Joystick $21.95
Famous Red Ball $23.95
Power Grip $21.95
Three-Way Deluxe $22.95
Atari/VIC Trackball $34.95
Apple Trackball $59.95

KRAFT
Atari Single Fire $13.00
Atari Switch Hitter $16.00
Apple Joystick $44.00
Apple Paddles $39.00

commodore

CBM 8032 $599
CBM64 ...$219
VIC 20 ....$99
CALL ON
Executive 64 Portable

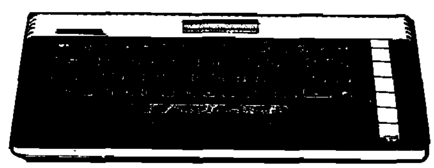
1520 Color Printer/Plotter $169.00
1525 80 Column Printer $219.00
1526 ....$319.00
1530 Datasette $69.00
1541 Single Disk Drive $249.00
1600 VIC Modem $59.00
1650 AD/AA Modem $89.00
1702 14" Color Monitor $249.00
Pet 64 ....$569.00
Pet 4032 ....$599.00
CBM 8032 ....$599.00
Super Pet $999.00
B128-80 ....$769.00
BX256-80 ....$969.00
2031 ....$299.00
4040 ....$699.00
8050 ....$949.00
8250 ....$1199.00
9060 ....$1999.00
9090 ....$2199.00
4023 ....$379.00
6400 ....$1399.00
64K Upgrade $269.00
Spell Master $149.00
Z-Ram $549.00
Silicon Office $749.00
The Manager $209.00
Soft Rom $129.00
Jinsam ....CALL
Call Result 64 ....$139.00

PROFESSIONAL SOFTWARE
Word Pro 2 Plus $159.00
Word Pro 3 Plus $189.00
Word Pro 4 Plus $279.00
Word Pro 5 Plus $279.00
InfoPro $179.00
Administrator $399.00
Power $79.00
Word Pro 64 Plus $65.00

CAROCO
for VIC 20/64

Light Pen $32.00
Cassette Interface $29.00
Parallel Printer Interface $69.00
3 Slot Expans. Interface (20) $32.00
6 Slot Expans. Interface (20) $79.00

ATARI HOME COMPUTERS



ATARI 600XL $149
ATARI 800XL $269
ATARI 1200XL ....CALL
ATARI 400 ....CALL
ATARI 800 ....CALL

1010 Program Recorder $74.00
1020 40 Col. Printer/Plotter $249.00
1025 80 Col. Printer $449.00
1027 Letter Quality Printer $299.00
1050 Disk Drive $379.00
1030 Direct Connect Modem ....CALL
CX30 Paddles $12.00
CX40 Joystick $8.00
CX42 Remote Joystick ....CALL
CX77 Touch Tablet $69.00
CX80 Trak Ball $49.00
CX85 Keypad $105.00
CX418 Home Manager $69.00
CX488 Communicator II $229.00
KX7098 Atari Accountant $209.00
KX7101 Entertainer $69.00
KX7102 Arcade Champ $75.00

ALIEN
Atari Voice Box $119.00
Apple Voice Box $149.00

MEMORY BOARDS
Axlon 32K Ram $59.00
Axlon 48K Ram $99.00
Axlon 128K Ram $299.00
Intec 32K Board $59.00
Intec 48K Board $85.00
Intec 64K Board (400 only) $99.00
Intec Real Time Clock $39.00

DISK & CART HOLDERS
Flip-n-File 10 ....$4.00
Flip-n-File 50 ....$19.00
Flip-n-File 400/800 Cart $19.00

PERCOM
AT 88-S1 ....$329.00
AT 88-A2 ....$269.00
AT 88-S2 ....$569.00
AT 88-S1 PD $469.00
AT 88-DDA $145.00
RFD 40-S1 ....$449.00
RFD 40-A1 ....$279.00
RFD 40-S2 ....$729.00
RFD 44-S1 ....$539.00
RFD 44-S2 ....$869.00
TX 99-S1 ....$279.00

RANA
1000 Atari Disk Drive $319.00

FLOPPY DISKS
MAXELL

MD-1 ....$29.00
MD-2 ....$44.00
FD-1 (8") $40.00
FD-2 (8" DS DD) $50.00

ELEPHANT

5¼" SS SD $18.50
5¼" SS DD $24.95
5¼" DS DD $29.95

VERBATUM

5¼" SS DD $26.00
5¼" DS DD $36.00

HEAD

Disk Head Cleaner $14.95

C.M.O. TOP 80

APPLE/FRANKLIN
1. Choplifter $27.00
2. Bank Street Writer $55.00
3. PFS: File $89.00
4. Visicalc $179.00
5. Home Accountant $55.00
6. Zaxxon $29.00
7. Most Amazing Thing $28.00
8. Visifile $189.00
9. Fathoms 40 ....$19.00
10. Deadline $35.00
11. PFS: Report $89.00
12. Zork III $29.00
13. Frogger $24.00
14. Facemaker $24.00
15. Snooper Troops #1 ....$32.00
16. Delta Drawing $35.00
17. Castle Wolfenstine $24.00
18. Wayout $29.00
19. Canyon Climber $19.00
20. Bandits $26.00

CBM 64
1. Word Pro 64 ....$65.00
2. Jumpman $29.00
3. Gorf (20/64) $14.95
4. Microspec Data Base 64 ....$69.00
5. Logo 64 ....$39.00
6. Microspec Gen. Ledger 64 ....$79.00
7. Zork III $29.00
8. Frogger (64) $23.00
9. Quick Brown Fox (20/64) $49.00
10. Shamus $29.00
11. Deadline $29.00
12. Assembler 64 ....$14.95
13. Zork II $29.00
14. 3-D Man $14.00
15. Protector $32.00
16. Starcross $29.00
17. Easy Mail 64 ....$14.95
18. Grave Robber $11.00
19. Wall Street $19.00
20. Trash Man $32.00

ATARI
1. Donkey Kong $39.00
2. Zaxxon $29.00
3. E.T. Phone Home $39.00
4. Miner 2049er $35.00
5. Dig Dug $33.00
6. Choplifter $29.00
7. Donkey Kong, Jr $39.00
8. Canyon Climber $25.00
9. Snooper Troops #2 ....$34.00
10. Word Wizard $59.00
11. Picnic Paranoia $34.00
12. Jumpman $29.00
13. Shamus $34.00
14. Letter Perfect $109.00
15. File Manager 800 ....$69.00
16. Preppie $24.00
17. Astro Chase $25.00
18. Blade/Black Hole $27.00
19. Pac Man $33.00
20. Baja Buggies $25.00
21. Crush, Crumble & Chomp $24.00
22. Wayout $27.00
23. Zork II $29.00
24. Visicalc $159.00
25. Atari Writer $49.00
26. Three Little Pigs $25.00
27. Upper Reaches of Apshai $16.00
28. Starbowl Football $24.95
29. Drelbs $26.00
30. Protector $34.00
31. Frogger $25.00
32. Lunar Leeper $24.00
33. Wizard of Wor $34.00
34. Kindercomp $21.00
35. Moon Shuttle $22.00
36. Home Accountant $55.00
37. Temple of Apshai $29.00
38. Spell Wizard $39.00
39. Nautilus $26.00
40. O'Riley's Mine $22.00

WEST
1-800-648-3311
In NV call (702)588-5654, Dept. 1115
P.O. Box 6689, Stateline, NV 89449
Order Status #: 588-5654

CANADA
1-800-268-4559
In Toronto call (416)828-0866, Dept. 1115
2505 Dunwin Ct., Unit 1B,
Mississauga, Ontario, Canada LSLITI
Order Status #: 828-0866

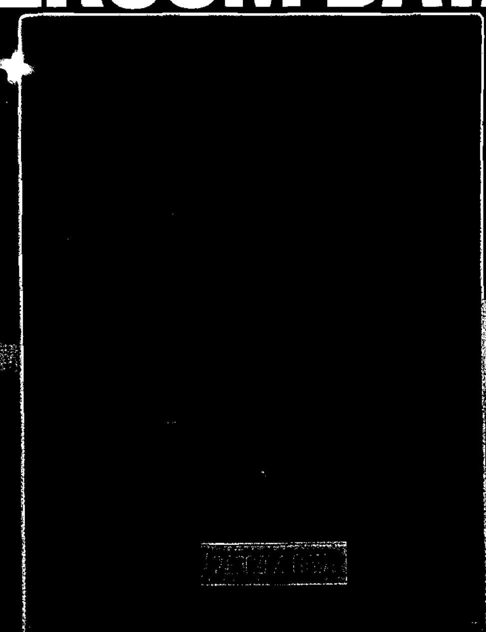
EAST
1-800-233-8950
In PA call (717)327-9575, Dept. 1115
477 E. Third St. Williamsport, PA 17701
Order Status #: 327-9576

VISA

CANADIAN ORDERS: All prices are subject to shipping, tax and currency exchange fluctuations. Call for exact pricing in Canada.
INTERNATIONAL ORDERS: All shipments outside the Continental United States must be pre-paid by certified check only. Include 3% (minimum $5.00) shipping and handling. EDUCATIONAL DISCOUNTS: Additional discounts are available to qualified Educational Institutions.
APO & FPO: Add 3% (minimum $5.00) shipping and handling.

ATARI COMPUTER OWNERS:

Pick the positively perfect, practical, peripheral package, from PERCOM DATA!



That's right ... the positively perfect PERCOM DATA 5¼" floppy disk drive with a BUILT-IN PRINTER-PORT, for your Atari® 400/800 is now available!

Until now, Atari computer owners who wanted to hook a printer to their computer had only one choice ... spend about $220 for an interface device. THOSE DAYS ARE OVER. PERCOM DATA has built a parallel printer-port right into its new AT88 PD model. Now you can add a quality disk drive system AND have a place to plug in a printer ... WITHOUT BUYING an interface.

The AT88 S1 PD™ disk drive operates in both single density (88K bytes formatted) and double density (176K bytes formatted).

What more could you want? NO INTERFACE ... a high quality PERCOM DATA disk drive ... AND a built-in PRINTER-PORT ... all with a price of $599.

Pick up a positively perfect PERCOM DATA disk drive, with printer-port ... pronto!

For the name of an authorized PERCOM DATA Dealer near you, call our **TOLL-FREE HOTLINE 1-800-527-1222 NOW,** or write for more information.

Perfectly Priced

$599.

PERCOM DATA
CORPORATION

Expanding Your Peripheral Vision

DRIVES • NETWORKS • SOFTWARE

11220 Pagemill Road, Dallas, Texas 75243 (214) 340-5800
1-800-527-1222

Introduction to Apple Graphics

by Phil Daley

If you ever have spent a long time programming an introductory or menu screen, then this month's screen editor will save you a lot of time. Also, it can be used to BSAVE a screen to be BLOADed into text screen page 2 enabling an on-line help screen with a simple POKE.

The Bar Graph Generator program is a very sophisticated charting program which can make attractive looking multi-colored high-resolution graphic displays easily and quickly. Also, if you have a hi-res screen dump, you can produce hard-copy printouts.

Perhaps you have written a program that you didn't save the latest version before running it, and somehow, DOS disappeared leaving you in the lurch with an inoperable disk drive. Ken and Ernest Gagnon present a technique for rebooting DOS without losing the program currently in memory. Remember where this tip is for that one chance in a thousand unexpected failure.

Our Apple columnist presents a look at a new, improved Applesoft from Beagle Bros. It may have just the features for which you've been looking.

Apple Screen Editor

by Phil Daley

Program Operation

After typing in and running the Apple listing for Screen Editor, the program asks if you wish to SAVE or LOAD a picture. Since you haven't saved any pictures to load, select 'S' for SAVE. The next question is whether you want to save a Text or Binary file. A binary save dumps the screen memory to disk. If you use it in another program you will be required to BLOAD to restore the picture. The text-file save routine creates a program of DATA statements that can be used from BASIC to restore the picture to the screen. Simply EXEC the file into memory and then SAVE it under a different name. You can use RENUMBER to change the line numbers and &Merge it into any other program.

The Atari and CoCo versions create strings of characters that can be printed to restore the screen picture. If anyone knows a method to PRINT flashing or inverse characters without preceding the character with that command, please let me know.

The Apple standard screen starts at 1024 and uses 1K of memory up to 2047. BASIC then starts at 2048. It is also possible to use the Page 2 text screen. This starts at 2048 and runs up to 3071. To use this screen as a help screen or additional information, it is necessary to move the BASIC program up to leave the screen area clear.

You can't PRINT to the second page, but you can POKE your characters directly to the screen memory, or else print them to page 1 and use a memory move to transfer them to page 2.

The 80-column cards for the old Apples use RAM on the card itself for the screen display. Many can display both 40 and 80 columns at the same time with the proper CRT connections. The IIe in mode 80 uses both the 40 column screen memory and additional memory on the auxiliary card. This is interlaced by the hardware during screen display.

Choosing 'L' for Load file, enter the name of a previously saved binary file. (If you have forgotten the name, press return for a 'CATALOG'.) After loading the file you will be left in the editor with the cursor in the upper left corner. The following control characters affect the program operation without changing the screen itself:

Character	Operation
RETURN	Cursor beginning of next line
CTRL-O	Cursor up one line
CTRL-L	Cursor down one line
Right arrow	Cursor right one space
Left arrow	Cursor left one space
CTRL-N	Normal mode
CTRL-I	Inverse mode
CTRL-F	Flashing mode
ESC	Save program
CTRL-@	Clear screen
CTRL-P	Dump screen to printer

All other keys print characters on the screen except the remaining control characters, which are discarded. The cursor is displayed as the inverse character of the current screen character, normal if the character is inverse or flashing. You can design fancy screens and save them for use in your programs.

Program Description

Lines 30-70 store the current character in X0 by reading the screen using SCRN( ) (see page 87 of the *Applesoft Programmer's Manual*). They also store the current mode (inverse, normal, flashing) in I0. Lines 90-130 print the new character in the proper mode on the screen at the cursor location.

Line 150 sets all the control keys. If you want to make other keys move the cursor, then change this line. Line 160 initializes the cursor to the upper left corner. Lines 170-250 get all the file information.

The main part of the program is from line 270 to line 530. The first thing the program does is to check that the cursor is within the normal screen boundaries and if it isn't, update the proper horizontal and vertical position variables so that it is. Lines 320-340 print the cursor on the screen inversed from the current value. Lines 350-370 are the keyboard input routine. Lines 400-500 are the control-key moves; they update the cursor without affecting the screen except for the '@', which clears the screen. Lines 520-530 actually print the character, update the position, and loop back for the next cycle through the main loop.

The next part of the program is the save routine. Line 550 restores the current character to the screen so that the cursor won't be saved. Line 580 does the BSAVE and returns. Lines 590-760 write a file similar to the one on page 77 of the DOS Manual. This save creates data statements to be read in a loop instead of the 1024 POKE statements described in the manual. You can save a picture several times during the drawing process if you wish. One modification might be to have it save a sequential name on each subsequent save; for instance: "PICTURE1", "'PICTURE2", ... etc.

The last part of the program is the screen dump. Line 800 restores the current character under the cursor. In line 810, change the "1" to the slot of your printer, if different. Lines 830-900 read all of the screen data into temporary arrays. Lines 920 to 1060 screen out the flashing characters and send the ASCII codes to the printer and line 1070 turns the printer off.

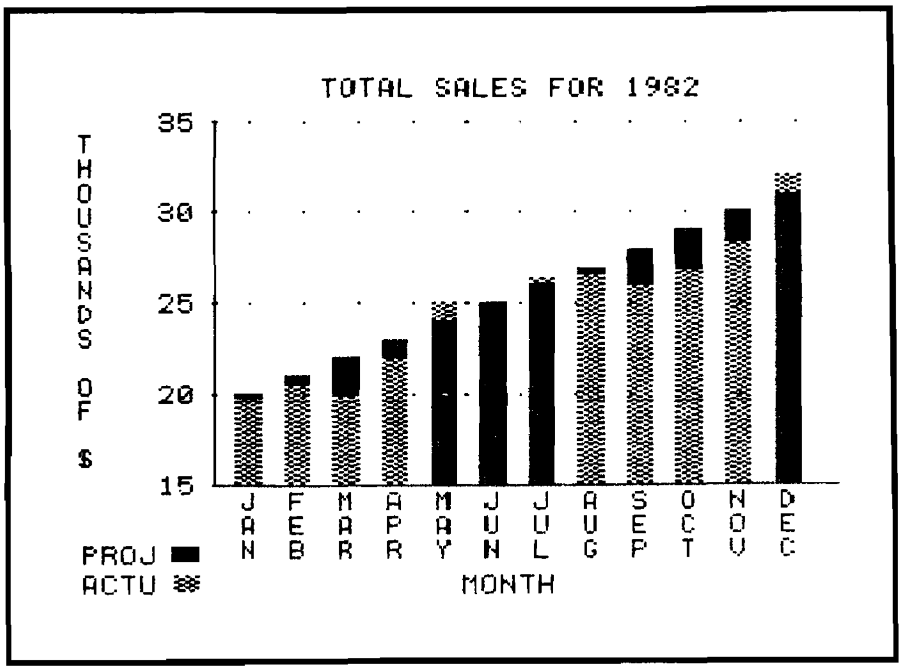
Apple Screen Editor Listing

```
10  TEXT : HOME : GOTO 150
20  REM
       READS SCREEN
30 XO =  SCRN( H,2 * V) + 16 *  SCRN( H,2 * V + 1)
40  IF XO < 64 THEN XO = XO + 64:IO = 1: GOTO 70
50  IF XO < 128 THEN IO = 2: GOTO 70
60 IO = 3
70  RETURN
80  REM
       PRINTS NEW CHAR ON SCREEN
90  IF INF = 1 THEN  INVERSE
100  IF INF = 2 THEN  FLASH
110  IF INF = 3 THEN  NORMAL
120  IF V = 23 AND H = 39 THEN H = H - 1
130  POKE CH,H: VTAB V + 1: PRINT  CHR$ (X1);: RETURN
140  REM
       INITIALIZE
150 CH = 36:DN = 12:CR = 13:BK = 8:FD = 21:IV = 9:
    NO = 14:FL = 6:UP = 15:ESC = 27:I1 = 3
160  DIM A%(350),B%(350),C%(350)
170 D$ =  CHR$ (13) +  CHR$ (4)
180 H = 0:V = 0
190  VTAB 10: INVERSE : PRINT "S";: NORMAL :
    PRINT "AVE OR ";: INVERSE :
    PRINT "L";: NORMAL : PRINT "OAD"
200  INPUT A$: IF  LEFT$ (A$,1) <  > "L" THEN 240
210  PRINT : PRINT "FILE NAME? <RETURN> FOR CATALOG"
220  INPUT A$: IF  LEN (A$) = 0 THEN
    PRINT  CHR$ (4)"CATALOG": GOTO 10
230  PRINT  CHR$ (4)"BLOAD"A$: GOTO 280
240  PRINT : PRINT : INVERSE : PRINT "P";: NORMAL :
    PRINT "ROGRAM OR ";: INVERSE :
    PRINT "B";: NORMAL : PRINT "INARY?"
250  INPUT P$:P$ =  LEFT$ (P$,1)
260  PRINT : PRINT "PROGRAM NAME?": INPUT A$
270  HOME
280  REM
       MAIN LOOP
290  IF H < 0 THEN H = 39:V = V - 1
300  IF H > 39 THEN V = V + 1:H = 0
310  IF V > 23 THEN V = 0
320  IF V < 0 THEN V = 23
330  GOSUB 30:X1 = XO
340  IF IO = 1 OR IO = 2 THEN INF = 3
350  IF IO = 3 THEN INF = 1
360  GOSUB 90
370 XN =  PEEK ( - 16384): IF XN < 127 THEN 370
380  POKE  - 16368,0
390 XN = XN - 128
400  REM
       CHECK FOR CURSOR MVT
410  IF XN > 27 THEN 540
420  IF XN = CR THEN X1 = XO:INF = IO:
    GOSUB 90:V = V + 1:H = 0: GOTO 290
430  IF XN = DN THEN X1 = XO:INF = IO:
    GOSUB 90:V = V + 1: GOTO 290
440  IF XN = BK THEN X1 = XO:INF = IO:
    GOSUB 90:H = H - 1: GOTO 290
450  IF XN = FD THEN X1 = XO:INF = IO:
    GOSUB 90:H = H + 1: GOTO 290
460  IF XN = UP THEN X1 = XO:INF = IO:
    GOSUB 90:V = V - 1: GOTO 290
470  IF XN = 27 THEN  GOSUB 580: GOTO 290
480  IF XN = 16 THEN  GOSUB 790: GOTO 290
490  REM
       CHECK FOR MODE CHANGE
500  IF XN = (IV) THEN I1 = 1
510  IF XN = FL THEN I1 = 2
520  IF XN = (NO) THEN I1 = 3
530  GOTO 370
540  REM
       PRINT CHAR & LOOP
550 X1 = XN:INF = I1: GOSUB 90
560 H = H + 1: GOTO 290
570  REM
       SAVE PROGRAM
580 X1 = XO:INF = IO: GOSUB 90
590  IF P$ = "P" THEN 610
600  PRINT D$"BSAVE"A$",A$400,L$400": RETURN
610 A = 4 * 256:B = A
620  PRINT D$"OPEN ";A$
630  PRINT D$"WRITE ";A$
640  PRINT LI;:LI = LI + 1
650  PRINT "FOR I =";A"TO";A + B
660  PRINT LI;:LI = LI + 1
670  PRINT "READ A:POKE I,A :NEXT"
680  FOR PLACE = A TO A + B
690 CO = CO + 1: IF CO = 10 THEN CO = 1
700  IF CO <  > 1 THEN  GOTO 730
710  PRINT : PRINT LINE;:LINE = LINE + 1
720  PRINT " DATA ";
730  IF PL = A + B THEN  PRINT  PEEK (PLACE): GOTO 760
740  PRINT  PEEK (PLACE);",";
750  IF CO = 9 THEN  PRINT  PEEK (PLACE + 1):PL = PL + 1
760  NEXT PLACE
770  PRINT : PRINT D$;"CLOSE ";A$
780  RETURN
790  REM
       PRINT SCREEN
800 X1 = XO:INF = IO: GOSUB 90
810  PRINT D$"PR#1"
820  PRINT : PRINT
830  FOR K = 0 TO 7
840 I = 1024 + 128 * K
850  FOR J = I TO I + 40
860 A%(J - 1024 - K * 128 + K * 40) =  PEEK (J)
870 B%(J - 1024 - K * 128 + K * 40) =  PEEK (J + 40)
880 C%(J - 1024 - K * 128 + K * 40) =  PEEK (J + 80)
890  NEXT
900  NEXT K
910 QQ = 40
920  FOR I = 1 TO 320
930 J = I - 1: IF A%(J) < 65 THEN A%(J) = A%(J) + 64
940  PRINT  CHR$ (A%(J));
950  IF I / QQ =  INT (I / QQ) THEN  PRINT
960  NEXT
970  FOR I = 1 TO 320
980 J = I - 1: IF B%(J) < 65 THEN B%(J) = B%(J) + 64
990  PRINT  CHR$ (B%(J));
1000  IF I / QQ =  INT (I / QQ) THEN  PRINT
1010  NEXT
1020  FOR I = 1 TO 320
1030 J = I - 1: IF C%(J) < 65 THEN C%(J) = C%(J) + 64
1040  PRINT  CHR$ (C%(J));
1050  IF I / QQ =  INT (I / QQ) THEN  PRINT
1060  NEXT : PRINT
1070  PRINT  CHR$ (4)"PR#0"
1080  RETURN
```

MICRO

BAR GRAPH GENERATOR

By Art Arizpe



This Applesoft program allows you to quickly create colorful bar graphs and histograms. The program supports hardcopy output and can save the graph in a text file for easy updating and editing.

Bar Graph requires:
Apple II, 48K and DOS. Optional Graphic Printer for Hardcopy

Bar Graph Generator allows you to generate professional quality bar graphs and histograms with a minimum of effort. There are several commercial packages available that also accomplish this, but when I found myself needing this capability I concluded that it would be cheaper, more challenging, and more enlightening if I wrote the software myself.

The program is written in Applesoft BASIC and requires 48K of RAM and one disk drive to operate. The program lets you save your graphs on disk for future retrieval and updating. A printer with graphics dump capability is also supported. Every effort has been made to crash-proof the program; all user input is tested for acceptability and the program rejects incorrect responses. All errors are trapped for further safe-guarding.

Using the Bar Graph Generator

Bar Graph Generator can be simply RUN from Applesoft BASIC or you can save it as the greeting program on a newly initialized disk so that it will run upon system booting. The program is menu-driven and self-prompting. The following options are available from the main menu:

E Enter values
L Load values
S Save graph
M Modify values
R Redraw graph
P Print graph
Q Quit program

Option E: Enter values

You can enter values for the bar graph by selecting option E from the main menu. You are asked for the number of points that you wish to enter for this graph. This number can be anywhere from 1 to 12. Twelve was chosen as the maximum to keep the graph from appearing cramped and at the same time allowing you to enter data in a monthly format for a full year. Of course, if you have more than 12 points to graph, you can enter and save more than one graph. You can also enter fewer than 12 points initially and add more values at a later time.

(Continued on page 92)

FOR YOUR APPLE II

Industry standard products at super saver discount prices

SOFTWARE

	List	SGC
ARTSCI		
Magicalc	$149.00	$ 99.00
Magic Window	149.00	99.00
DBase (Apple)	695.00	475.00
BRODERBOND		
Payroll	$395.00	$295.00
Arcade Machine	59.95	39.95
Serpentine	34.95	24.95
Galaxy Wars	24.95	20.95
DATAMOST		
Aztec	$ 39.95	$ 27.95
Casino "O"	39.95	27.95
Worlds	29.95	20.95
Mating Zone	29.95	20.95
Pandoras Box	29.95	20.95
Swashbuckler	34.95	24.95
Roundabout	29.95	20.95
INFOROM		
Zork 1, 2, 3	$ 39.95	$ 27.95
Planet Fall	49.95	32.95
Starcross	39.95	27.95
PEACHTREE		
Accounts Payable	$400.00	$295.00
Accounts Receivable	400.00	295.00
General Ledger	400.00	295.00
Inventory	400.00	295.00
SENTIENT		
Congo	$ 34.95	$ 24.95
Cyborg	34.95	24.95
Goldrush	34.95	24.95
SIRIUS		
Bandits	$ 34.95	$ 24.95
Epoch	34.95	24.95
Fly Wars	29.95	20.95
Beer Run	29.95	20.95
Outpost	29.95	20.95
Snake Byte	29.95	20.95
Sneakers	29.95	20.95
Joy Port	49.95	37.95
STRATIGIC SIMULATIONS		
Bomb Alley	$ 59.95	$ 42.95
Computer Combat	59.95	42.95
Ambush	59.95	42.95
Quarter Back	39.95	27.95
Germany 1985	59.95	42.95
The Warp Machine	39.95	27.95
Gettysburg	59.95	42.95
DANA		
Series Baseball	$ 29.95	$ 20.95
Wiz Plus	39.95	27.95
T.G. Joystick	59.95	45.00
Select-A-Port	59.95	45.00

SPECIAL AND NEW

BASIS COMPUTER

128K Memory • 80 Column Card • C.P.M.3.0, DOS Perfect Calc • Perfect Filer • Perfect Speller • Perfect Writer • 2 Disk Drives • Monitor

Special $2295.00

FRANKLIN ACE 1000

64K, 1 Disk Drive • 64K Expand-A-Ram • Magicalc

Special $1595.00

64K Expand-A-Ram	$ 225.00
128K Expand-A-Ram	$ 275.00
P.S. Buffer Card Stand Alone	$ 155.95
Graphitti Card (Specify Printer)	$ 89.00

PRINTERS

NEC 8023	$ 395.00
NEC 8023 with Parallel Interface/Cable	$ 449.00
OKI 82A 80 col./120 cps.	$ 439.00
OKI 83A 136 col./120 cps.	$ 695.00
Brother Letter Quality	$ 889.00

ACCESSORIES

Disk Drives with Controller (Apple)	$ 225.00
VersaCard	$ 169.00
With Graphics	$ 203.00
P.S. Buffer Card 16K Buffer	$ 105.95
P.S. Buffer Card with Graphics & Cables	$ 189.95
16K Card	$ 65.00
Floppy Discs Box of Ten	$ 25.00
With Order	$ 19.90
PRT 1 Parallel with Cable	$ 79.00

BUSINESS SPECIAL

Includes 80 Column Z-80 Card • 16K Card • Wordstar • Supercalc

Special **$825.00**

D.O.S. Programming SPECIAL (Apple II)

80 Column Card • Magicalc • Magic Window • Magic Speller • 16K Card

Special **$449.00**

3-CARD SYSTEM

80 Column 16K Card • Z-80 Card

Special **$349.95**

MODEMS

Smartmodem 300

Special **$229.00**

Smartmodem 1200

Special **$545.00**

D-Cat Modem

Special **$164.95**

MONITORS

Amdex Amber	$159.95
USI Green	$159.95
USI Lo/Res Color	$299.95

80 COLUMN CARDS

Smarterm 2	$129.00
Videx Video Term.	$215.95
Ultra Term.	$289.95

Z-80 CARDS

Microsoft Card	$229.95
Apple IIE, 64K, 80 Column	$ 99.00

EDUCATIONAL SOFTWARE

Spelling/Reading Primer	$ 29.95
Perception	$ 20.95
Counting Bee	$ 23.95
Word Scrambler	$ 15.95
Fractions	$ 36.95
Intro. Algebra	$ 21.95
Frenzy/Flip Flop	$ 25.95
Wordstar	$305.95
SuperCalc	$175.00
VisaCalc	$179.00
Spellstar	$175.00

All equipment shipped factory fresh. Manufacturers' warranties included. California customers add 6½% tax. Include payment by personal check, money order, or cashier's check with order and SGC will pay shipping charge. Call for amount of shipping charge when paying by credit card.

All items are normally in stock

(415) 490-3420

. . . And we'll be here to help after you receive your order. Feel free to call the SGC Technical Staff for assistance.

SGC

The mail order specialists

342 Quartz Circle, Livermore, CA 94550

Next you are asked whether you wish to enter one or two vertical values for each horizontal point. This allows you to generate bar graphs with single or double bars. For each point you wish to enter, you are asked to provide the horizontal label and one or two vertical values. The program displays only the first three characters of the horizontal labels. Because of space limits on the screen, you should also keep your numeric vertical values to less than five characters.

The program then asks you for the low-scale value. This number must be less than the lowest vertical value that you entered. The program also prompts you for the high-scale value and for the increment or step value on the vertical axis. Because of round-off error you should keep the increment to less than three decimal places.

You are then asked for the vertical and horizontal axis labels. You can enter any string up to 255 characters long for these, but the horizontal label is truncated to the first 30 characters of your response. The vertical label is truncated to the first 16 characters. These labels are optional but make the graph easier to understand. You can also enter a title for the graph, which is, again, truncated to 30 characters. If you have specified two vertical values per horizontal entry, you are asked to give labels to the two vertical variables. These labels appear as a legend in the lower left corner of the graph and are truncated to four characters. You then select bar colors for your graph and the program draws the bar graph. Once the graph is complete, pressing any key will return you to the main menu.

Option L: Load values

Selecting option L from the main menu allows you to read in values for a graph that was saved from option S (see below). These values are read from a text file and you are asked to provide the proper file name. This option is useful when you wish to update a graph with new information.

Option S: Save graph

Option S allows you to save the bar graph as a hi-res screen binary file or as a values text file. In each case you are asked to provide a file name. Saving the graph as a binary file allows you to display it from another program; saving the graph as a text file allows easy updating and saves considerable disk space. A typical graph takes only two sectors as a text file but occupies 34 sectors as a binary file. Note that binary files are saved with a .PIC suffix and that text files are saved with a .DATA suffix, which allow you to give the binary file and the text file the same name. When you are asked to provide a file name (as in option L above) you should not enter the .PIC or .DATA suffix.

Option M: Modify values

Option M takes you to the Modify sub-menu. You can modify vertical and horizontal values. The program asks which point (entry) you wish to modify. You can also select to modify the various graph labels and the scale limits and increment. The program tells you what the current value is and asks for the new value. Pressing return without entering a new value leaves the current value unchanged. After any editing, you can choose to continue modifying or to return to the main menu.

Option R: Redraw graph

Selecting option R allows you to redraw the graph. You should redraw the graph after any editing of values so that the hi-res screen will reflect these changes. The program reminds you to do this if you forget.

Option P: Print graph

If you have a printer with graphics capabilities, option P lets you print the bar graph. The program asks you to ensure that the printer is on. You will need to modify lines 3440-3470 to accommodate your particular printer's graphics software. Optionally, you can use a screen-dump program to print the saved hi-res screen.

Option Q: Quit program

Select option Q to exit the program when you are done.

Program Organization and Operation

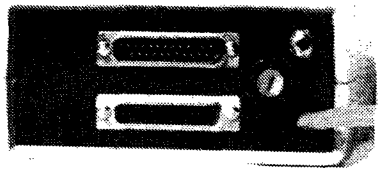
The program is written in a modular form with comments indicating where the various subroutines exist. Since the program outputs characters to the hires screen, I created a shape table for all the characters that are normally available from the Apple's keyboard

(Continued on page 94)

P.O. Box 4364
Flint, Michigan 48504
(313) 233-5731
(313) 233-3125

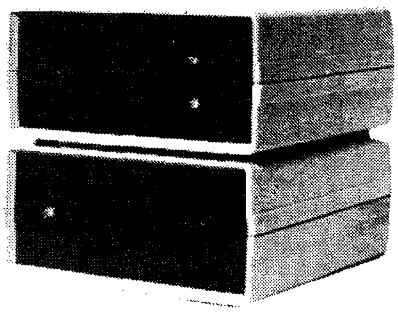
ZANIM SYSTEMS

HOME CONTROL AT YOUR FINGERTIPS



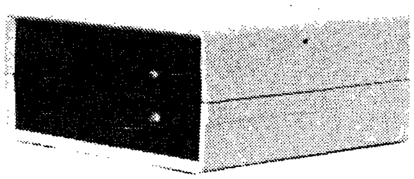
The ZCM-1 is the Master Control module that provides the interface between your computer and our line of Zanim Application Modules. Up to 15 Application Modules can be piggy-backed to the ZCM-1 Master Control module. The ZCM-1 is compatable with any standard RS-232 (serial) interface. A special Master Control module, the ZCM-1V is available for the VIC-20 and Commodore-64 computers.

*The ZCM-1V is available for VIC-20 and C-64 users.



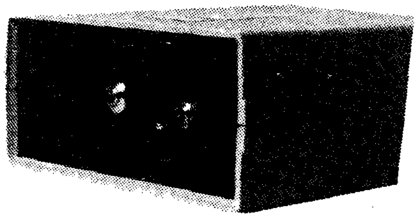
ZAM-1 is the home control interface module that provides a fully versatile computer controlled environment. ZAM-1 can control up to 256 different lamps and appliances in your home or business providing you with an effective and easy to implement energy management and electric control system. No special modifications are necessary to your building as all control signals are sent over your existing wiring. ZAM-1 can be programmed in BASIC or optional home control software is available. ZAM-1 requires one ZCM-1 Master Control module.

*The ZCM-1/ZCM-1V Master Control module is required to use the ZAM-1 Home Control module.



The ZAM-2 allows your computer to continuously monitor up to 15 different doors or windows around your home or business. ZAM-2 is a basic building block in a complete computer controlled home security system. With our ZAM-1 Home Control module, you can have a fully integrated security and environment control system. Upon an intrusion, your computer can take the action most appropriate, whether that is to ring an alarm bell, flash all the lights around your home, or dial the police.

*The ZCM-1/ZCM-1V Master Control module is required to use the ZAM-2 Security module.



The ZAM-3 is a complete telephone answering and dialing system. It is capable of taking the phone off-hook and dialing a number under computer control or of answering the phone when it rings. With the ZAM-1 Home Control module and the ZAM-2 Security module, the ZAM-3 Phone Dialer module can be integrated into a complete home or business security/monitoring system. Applications include security, auto phone dialing, and computer-answering systems.

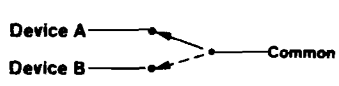
*Pulse dialing option is available as ZAM-3P.

*The ZCM-1/ZCM-1V Master Control module is required to use the ZAM-3/ZAM-3P Phone Dialer module.

SERIAL OR PARALLEL (CENTRONICS) PORT SWITCHER

DOES YOUR COMPUTER LOOK LIKE THIS?

A PORT SWITCHER NOW AVAILABLE FOR YOUR COMPUTER (ZSW1)



P.O. BOX 4364
Flint, Michigan 48504
(313) 233-5731
(313) 233-3125

☐ Please send me more information or catalogue!

Name ____________________

Company ____________________

Address ____________________

City __________ State ________ Zip ________

What Make/Model Computer do you own?

____

(sorry, no lower case). This table is best entered from the monitor by following the hex dump that I have provided. The shape table should be saved as BAR.GRAPH.CHAR, on the same disk as the program. This can be done by typing BSAVE BAR.GRAPH.CHAR, A$6000,L$373 <return>. The program automatically finds a safe address for the shape table and BLOADs it in.

My printer is a C. Itoh Prowriter controlled by a Prometheus GRAPHITTI card in slot 1. The code in lines 3440-3470 is set up for this printer/interface card combination, so you will need to change this to fit your system.

I have tried to anticipate some of the more common disk errors. The routine at lines 3170-3310 handles these errors and thus prevents the program from crashing. As a last resort, if an error occurs that is not handled by this routine, the program prints the line number at which the error occurred and terminates gracefully. The program also traps CTRL-C, so pressing CTRL-C will not stop the program. Instead, pressing CTRL-C while the program is running will return you to the main menu. There are two ways to exit the program: by pressing RESET or by selecting option Q from the main menu.

This error-handling routine can probably be modified for use within your own programs. Proper handling of errors is important, especially if you will not be the only user of your programs. As I mentioned before, Bar Graph Generator checks all keyboard input before proceeding. This may make the program a bit longer but the result is a virtually crash-proof program.

Another section of the program that you may find useful is the routine that outputs a string to the hi-res screen (lines 330-410). This routine scans the string, extracts each character, and determines the proper shape to DRAW. The variable V is a flag, which when equal to one causes the string to be output in a vertical format.

Possible Modifications

Most of the program deals with accepting and editing data; the section that draws the bars is quite small. One possible modification would be to allow plotting of points and trend plots. You might also want to list the data to the screen or a printer in a tabular format. It should be possible to dump the graph to a printer (even if the printer does not have graphics capabilities) by using character graphics.

Another possibility is to convert any program that you may already have to allow its output to be viewed with Bar Graph Generator. A text file that contains information in the format that Bar Graph Generator expects can be loaded as a values text file and graphed. The data stored is quite simple (refer to lines 3020-3060).

These are just a few of the possible modifications and expansions to the program and I would like to hear from any of you who try them.

Art Arizpe is an electrical engineer employed by Texas Instruments as a MOS integrated circuit designer. He does most of his programming in BASIC and 6502 assembly language but has also worked in FORTRAN and Pascal. His current interests include computer-aided design, graphics, and music. You may contact Mr. Arizpe at 857 B. Dublin Dr., Richardson, TX 75080.

Listing 1

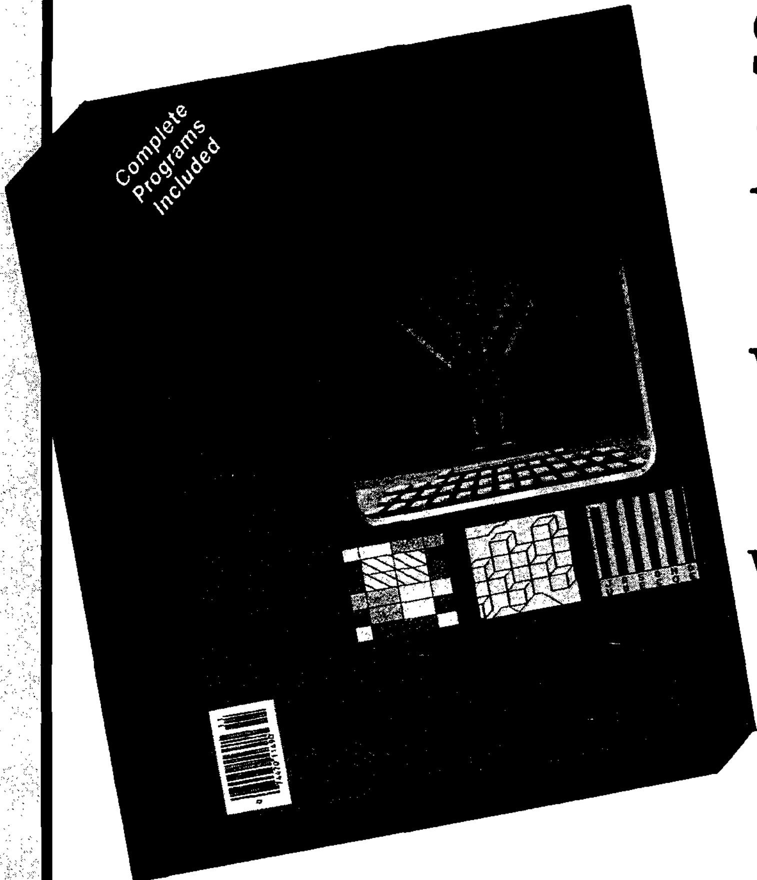
```
01  REM ************************
02  REM * BAR GRAPH GENERATOR  *
03  REM *     Written by       *
04  REM *     Art Arizpe       *
05  REM *       09/82          *
06  REM *  Copyright (C) 1983  *
07  REM *    by MICRO Ink      *
08  REM ************************
70 D$ =  CHR$ (4): DIM Y(2,12),X$(12)
80  PRINT D$"NOMONC,I,O": ONERR  GOTO 3170
90 HM =  PEEK (115) +  PEEK (116) * 256: IF  PEEK (HM) = 66 THEN 120
100 ADR = HM - 900: PRINT D$"BLOAD BAR.GRAPH.CHAR,A"ADR: HIMEM: ADR
110  POKE 232,((ADR / 256) -  INT (ADR / 256)) * 256:
       POKE 233, INT (ADR / 256)
120  SCALE= 1: ROT= 0: GOTO 3520
130  REM **********************
140  REM *     SOLID BARS     *
150  REM **********************
160  FOR J = 0 TO 9: HCOLOR= C1: HPLOT H + J,151 -
       TP TO H + J,150 - BT: NEXT : RETURN
170  REM **********************
180  REM *    SHADED BARS     *
190  REM **********************
200 L =  INT ((153 - BT) / 2) * 2: HCOLOR= C2:
       IF BT <3 THEN L = 151
210  FOR J = 151 - TP TO L STEP 2
220  HPLOT H,J: HPLOT H + 1,J: HPLOT H + 4,J:
       HPLOT H + 5,J: HPLOT H + 8,J: HPLOT H + 9,J: NEXT
230  FOR J = 152 - TP TO L STEP 2
240  HPLOT H + 2,J: HPLOT H + 3,J: HPLOT H + 6,J:
       HPLOT H + 7,J: NEXT : RETURN
250  REM **********************
260  REM *     HOR. LABEL     *
270  REM **********************
280 XS = H + 2:YS = 160:ST$ = X$(I): HCOLOR= 3: GOSUB 330
290  RETURN
300  REM **********************
310  REM *   OUTPUT STRING    *
320  REM **********************
```

(Continued on next page)

Listing 1 *(continued)*

```
330  FOR J = 1 TO  LEN (ST$)
340  IF ST$ = "" THEN 410
350  DRAW  ASC ( MID$ (ST$,J,1)) -30 AT XS,YS
360  IF V = 1 THEN 390
370 XS = XS + 7
380  GOTO 400
390 YS = YS + 9
400  NEXT
410  RETURN
420  REM *********************
430  REM *  FIND DATA LIMITS  *
440  REM *********************
450 LB = 1E38:UB =  - 1E38: FOR I = 1 TO SIZE: FOR J = 1 TO P
460  IF Y(J,I) > UB THEN UB = Y(J,I)
470  IF Y(J,I) <LB THEN LB = Y(J,I)
480  NEXT : NEXT
490  RETURN
500  REM *********************
510  REM *    ENTER VALUES    *
520  REM *********************
530  HOME : VTAB 2: HTAB 15: PRINT "ENTER DATA":
       PRINT : PRINT : POKE 34,3
540 Q$ = "      "
550  IF SIZE = 0 THEN MX = 12:P = 0: GOTO 670
560  IF SIZE = 12 THEN  HOME : VTAB 11: PRINT
       "YOU ALREADY HAVE 12 HORIZONTAL ENTRIES.":
       VTAB 24: HTAB 7: PRINT "PRESS ANY KEY TO CONTINUE ";:
       GET A$: HOME : VTAB 10: GOTO 580
570  VTAB 11: HTAB 5: PRINT "(A) ADD POINTS TO EXISTING GRAPH"
580  PRINT : HTAB 5: PRINT "(C) CLEAR CURRENT VALUES AND START
       NEW GRAPH"
590  PRINT : HTAB 5: PRINT "(M) RETURN TO MAIN MENU"
600  VTAB 24: HTAB 15: PRINT "WHICH ? ";: GET A$:
       IF  ASC (A$) <31 THEN 620
610  PRINT A$;
620  IF A$ < > "A" AND A$ < >
       "C" AND A$ < > "M" THEN 600
630  HOME : IF A$ = "C" THEN SIZE = 0:DRWN = 0:
       N = 0:P = 0:LO = 0:HI = 0:INC = 0:
       V$ = "":H$ = "":TITLE$ = "":Y1$ = "":Y2$ = "":
       FOR I = 1 TO 2: FOR J = 1 TO 12:
       Y(I,J) = 0: NEXT : NEXT : GOTO 540
640  IF A$ = "M" THEN 3490
650 MX = 12 - SIZE:DRWN = 0
660 Q$ = "TO ADD "
670  PRINT "ENTER NUMBER OF POINTS "Q$"
       (MUST BE FROM 1 - "MX") : ";: INPUT "";T$
680  IF T$ = "" THEN 3520
690  IF  VAL (T$) <1 OR  VAL (T$) > MX THEN
       VTAB  PEEK (37) - 1: CALL -868: GOTO 540
700 N =  VAL (T$)
710  IF P < > 0 THEN 760
720  PRINT : INPUT "DO YOU WANT 1 OR 2 VERTICAL VALUES
       PER HORIZONTAL POINT? ";T$
730  IF  VAL (T$) <1 OR  VAL (T$) > 2 THEN
       VTAB  PEEK (37) - 2: CALL  -868: GOTO 720
740 P =  VAL (T$)
750 LB = 1E38:UB =  - 1E38
760  PRINT : PRINT "ENTER YOUR ";N;" POINTS :": PRINT
770  FOR I = SIZE + 1 TO SIZE + N
780  PRINT "# ";I;: INPUT ",        HOR. LABEL: ";X$(I)
790 X$(I) =  LEFT$ (X$(I),3)
800 Q$ = "1ST": IF P = 1 THEN Q$ = "   "
810  FOR J = 1 TO P
820  PRINT "     ";Q$;: INPUT " VERT. VALUE: ";T$
830  IF T$ = "" OR ( STR$ ( VAL (T$)) < > T$)
       OR  LEN (T$) > 5 THEN VTAB PEEK (37): GOTO 820
840 Y(J,I) =  VAL ( RIGHT$ (T$,5))
850  IF Y(J,I) > UB THEN UB = Y(J,I)
860  IF Y(J,I) <LB THEN LB = Y(J,I)
870 Q$ = "2ND": NEXT
880  PRINT : NEXT
890  HOME : VTAB 6: PRINT "ENTER THE LOW, HIGH, AND
       STEP VALUES FORTHE VERTICAL SCALE :": PRINT
900  PRINT "LOW SCALE VALUE (MUST BE <"LB") ":
       INPUT T$: IF  STR$ ( VAL (T$)) < > T$
       OR  VAL (T$) >  = LB THEN  VTAB  PEEK (37) - 1:
       CALL  - 868: GOTO 900
910 LO =  VAL (T$)
920  PRINT : PRINT "HIGH SCALE VALUE (MUST BE >= "UB") ":
       INPUT T$: IF  STR$( VAL (T$)) < > T$
       T$ OR  VAL (T$) <UB THEN  VTAB  PEEK (37) - 2:
       CALL  - 868: GOTO 920
930 HI =  VAL (T$)
940  PRINT : INPUT "SCALE STEP VALUE ? ";T$:
       IF  STR$ ( VAL (T$)) < > T$ OR T$ = "0" OR
       VAL (T$) > HI - LO THEN  VTAB  PEEK (37) - 1:
       CALL  - 868: GOTO 940
950 INC =  VAL (T$)
960  HOME : VTAB 6: INPUT "ENTER THE VERTICAL AXIS LABEL : ";T$:
       IF T$ = "" THEN 980
970 V$ =  LEFT$ (T$,16)
980  PRINT : INPUT "ENTER THE HORIZONTAL AXIS
       LABEL : ";T$: IF T$ = "" THEN 1000
990 H$ =  LEFT$ (T$,30)
1000  PRINT : INPUT "ENTER THE GRAPH TITLE : ";T$:
       IF T$ = "" THEN 1020
1010 TITLE$ =  LEFT$ (T$,30)
1020  IF P = 1 THEN 1070
1030  PRINT : INPUT "ENTER THE 1ST Y-VALUE LEGEND LABEL : ";T$:
       IF T$ = "" THEN 1050
1040 Y1$ =  LEFT$ (T$,4)
1050  PRINT : INPUT "ENTER THE 2ND Y-VALUE LEGEND LABEL : ";T$:
       IF T$ = "" THEN 1070
1060 Y2$ =  LEFT$ (T$,4)
1070 SIZE = SIZE + N
1080  IF SIZE = 0 THEN  HOME : PRINT  CHR$ (7); CHR$ (7):
       VTAB 12: HTAB 4: PRINT
       "YOU HAVE NOT ENTERED ANY VALUES !": GOTO 3480
1090  REM *********************
1100  REM *    DRAW GRAPH     *
1110  REM *********************
1120 DRWN = 1
1130  HOME : VTAB 10: PRINT "ENTER BAR COLOR : ": PRINT
1140  PRINT "G - GREEN, B - BLUE, W - WHITE,
       V - VIOLET, O -ORANGE": PRINT
1150  IF P < > 1 THEN 1180
1160 I = 3: VTAB 17: PRINT "WHICH COLOR FOR Y VALUE BAR ? ";:
       GET A$: PRINT A$: IF  ASC (A$) <31 THEN 1150
1170  GOTO 1220
1180  FOR I = 1 TO P
1190  ON I GOTO 1200,1210
1200  VTAB 17: PRINT "WHICH COLOR FOR 1ST Y VALUE BAR ? ";:
       GET A$: PRINT A$: GOTO 1220
1210  VTAB 19: PRINT "WHICH COLOR FOR 2ND Y VALUE BAR ? ";:
       GET A$: PRINT A$
1220  IF A$ = "G" THEN C2 = 1: GOTO 1290
1230  IF A$ = "B" THEN C2 = 6: GOTO 1290
1240  IF A$ = "W" THEN C2 = 3: GOTO 1290
1250  IF A$ = "V" THEN C2 = 2: GOTO 1290
1260  IF A$ = "O" THEN C2 = 5: GOTO 1290
1270  VTAB  PEEK (37): CALL  - 868: IF I = 3 THEN 1150
1280  GOTO 1190
1290  IF I = 1 THEN C1 = C2
1300  IF P = 1 AND I = 3 THEN C1 = C2: GOTO 1320
1310  NEXT
1320  HGR2 : HCOLOR= 3
1330 V = 0:XS = (180 -  LEN (TITLE$) * 7) / 2 + 70
1340 YS = 6:ST$ = TITLE$: GOSUB 330
1350  REM *********************
1360  REM *     DRAW AXES     *
1370  REM *********************
1380  HPLOT 50,151 TO 279,151
1390  HPLOT 50,151 TO 50,16
1400 M = (HI - LO) / INC:N = HI + INC
1410  FOR I = 16 TO 151 STEP 135 / M
1420  HPLOT 49,I TO 51,I: FOR J = 63 TO 279 STEP 20:
       HPLOT J,I: NEXT
1430 XS = 35 -  LEN ( STR$ (INC)) * 5
1440 N =  INT ((N - INC) * 100 + .5) / 100
1450 YS = I + 3:ST$ =  STR$ (N): GOSUB 330
1460  NEXT
1470 V = 1
1480 YS = (160 -  LEN (V$) * 9) / 2 + 10
1490 XS = 0:ST$ = V$: GOSUB 330
1500 V = 0
1510 XS = (180 -  LEN (H$) * 7) / 2 + 70
1520 YS = 191:ST$ = H$: GOSUB 330
1530  IF P = 1 THEN 1620
1540  HCOLOR= C2
1550  FOR J = 185 TO 190 STEP 2: HPLOT 35,J: HPLOT 36,J:
       HPLOT 39,J: HPLOT 40,J: HPLOT 43,J: HPLOT 44,J:
       HPLOT 37,J + 1: HPLOT 38,J + 1: HPLOT 41,J + 1:
       HPLOT 42,J + 1: NEXT
1560  HCOLOR= C1: FOR I = 174 TO 179: HPLOT 34,I TO 44,I: NEXT
1570  HCOLOR= 3:XS = 2:YS = 191:ST$ = Y2$: GOSUB 330
1580 XS = 2:YS = 180:ST$ = Y1$: GOSUB 330
```

(Continued on page 97)



Subscribe to MICRO... Save 20% and we'll send you a BONUS GIFT with your subscription!

Fill out the attached card and mail today!

CHOOSE FROM:

- ★ Best of MICRO Vols. 2 and 3
  An $18.00 Value — FREE!

or

- ★ MICRO Calc for Apple, Atari, TRS-80, or C64
  A $14.95 Value — FREE!

or

- ★ What's Where in the Apple
  A $24.00 Value for Only $12.00

TWO NEW
FROM

ALSO Don't Miss
These Great MICRO Publications

A Feast Of Computing Ideas

Use These Postage-Paid Cards To Order

SAVE 20%!

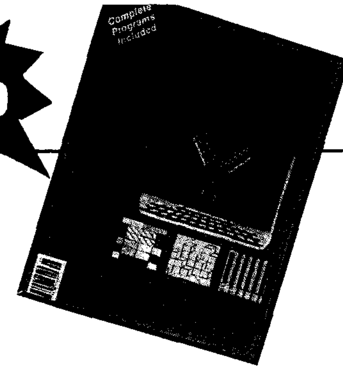
Use This Postage-Paid Card To Subscribe... Or Call Toll-Free

1-800-345-8112

(In PA 1-800-662-2444)

(MasterCard and VISA Accepted)

MICRO™ SAVE 20%



☐ YES! Enter my subscription to MICRO for just $24.00! and with it send me the BONUS GIFT I've checked.

☐ **Best of MICRO Volumes 2 *and* 3.** An $18.00 Value — **FREE!**

☐ **MICRO Calc** for ☐ Apple ☐ Atari ☐ TRS-80 ☐ C64
A $14.95 Value — **FREE!**

☐ **What's Where in the Apple** at Half Price! A $24.00 Value — **Only $12.00!**

Please rush my subscription and the BONUS GIFT I've checked to:

Name ______________________

Address ______________________

City ____________ State ________ Zip ________

(Allow 6-8 weeks for delivery)

I'm paying by: ☐ Check ☐ MO ☐ VISA ☐ MC

Total Enclosed: $ ____________

Visa/MC # ____________

Exp. Date: ____________

BOOKS MICRO!

MICRO™'s Newest 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 @ **$19.95**

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

Please rush the MICRO Books I've checked above to:

Name ______________________

Address ______________________

City ____________ State ________ Zip ________

(Allow 6-8 weeks for delivery)

MA residents add 5% sales tax.

I'm paying by: ☐ Check ☐ MO ☐ VISA ☐ MC

Total Enclosed: $ ____________
(Add $2.00 s/h per book.)

Visa/MC # ____________

Exp. Date: ____________

MICRO™'s Best Sellers

The Best of MICRO

Make your computer reference library complete by adding the "Best of MICRO" to your present book collection. Each volume full of articles and programs pulled from the pages of MICRO. Excellent software at a very low price!

☐ BOM Volume 2 @ $8.00
☐ BOM Volume 3 @ $10.00

What's Where in the Apple

Revised new addition with text added to the original atlas and gazetter. All Apple users will find this book helpful in understanding their machine and essential for mastering it.

☐ What's Where in the Apple @ $24.95
☐ The Guide @ $9.95

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)

MA residents add 5% sales tax.

I'm paying by: ☐ Check ☐ MO ☐ VISA ☐ MC

Total Enclosed: $ ____________
(Add $2.00 s/h per book.)

Visa/MC # ____________

Exp. Date: ____________

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY CARD
FIRST CLASS PERMIT NO. 20, AMHERST, NH

POSTAGE WILL BE PAID BY ADDRESSEE

MICRO™
10 Northern Blvd.
P.O. Box 6502
Amherst, NH 03031

SAVE 20%!!!!

Subscribe to MICRO

Use This Postage Paid Card to Orde
the Next 12 Issues
of MICRO and SAV
$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 O Computin Ideas...

New Boo From MICR

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

Order These

Best Sellers From MICR

Listing 1 *(continued)*

```
1590  REM *********************
1600  REM *     DRAW BARS     *
1610  REM *********************
1620 V = 1:K1 =  INT (220 / SIZE):K2 = 135 / (HI - LO)
1630  IF  INT (K1 / 2) <> K1 / 2 THEN K1 = K1 + 1
1640  FOR I = 1 TO SIZE:H = 58 + (I - 1) * K1:
        Y1 = (Y(1,I) - LO) * K2:Y2 = (Y(2,I) - LO) * K2
1650  IF P = 2 THEN 1670
1660  IF Y1 <135 AND Y1 > 0 THEN 1680
1670  IF Y1 > 135 OR Y2 > 135 OR Y1 <0 OR
        Y2 <0 THEN  TEXT : HOME : PRINT CHR$(7): VTAB 12:
        HTAB 7: PRINT "ERROR: VALUE FOR POINT # "I: HTAB 7:
        PRINT "EXCEEDS SCALE LIMITS.": PRINT  CHR$ (7): GOTO 3480
1680  IF P = 1 THEN TP = Y1:BT = 0: GOSUB 160: GOSUB 280: GOTO 1710
1690  IF Y1 > Y2 THEN TP = Y2:BT = Y1 - 1: GOSUB 160:
        TP = Y2:BT = 0: GOSUB 200: GOSUB 280: GOTO 1710
1700 TP = Y2:BT = Y1: GOSUB 200:TP = Y1:BT = 0: GOSUB 160: GOSUB 280
1710  NEXT
1720  GET A$: GOTO 3520

1730  REM *********************
1740  REM *   MODIFY VALUES   *
1750  REM *********************
1760  TEXT : HOME
1770  IF SIZE = 0 THEN  PRINT  CHR$ (7); CHR$ (7): VTAB 12:
        HTAB 4: PRINT "YOU HAVE NOT ENTERED ANY VALUES !": GOTO 3480
1780  HTAB 13: PRINT "MODIFY VALUES": POKE 34,2
1790  VTAB 8: HTAB 8: PRINT "(V) MODIFY VERTICAL VALUES": PRINT
1800  HTAB 8: PRINT "(H) MODIFY HORIZONTAL VALUES"
1810  PRINT : HTAB 8: PRINT "(L) MODIFY SCALE LABELS"
1820  PRINT : HTAB 8: PRINT "(S) MODIFY SCALE LIMITS"
1830  PRINT : HTAB 8: PRINT "(M) RETURN TO MAIN MENU"
1840  VTAB 24: HTAB 16: PRINT "WHICH? ";: GET A$:
        IF  ASC (A$) <31 THEN 1840
1850  PRINT A$;
1860  IF A$ <> "S" AND A$ <> "V" AND A$ <> "L" AND
        A$ <> "M" AND A$ <> "H" THEN 1840
1870  IF A$ = "S" THEN 1950
1880  IF A$ = "V" THEN 2440
1890  IF A$ = "H" THEN 2620
1900  IF A$ = "L" THEN 2180
1910  IF A$ = "M" THEN 3520

1920  REM *********************
1930  REM * MODIFY SCALE LIMITS*
1940  REM *********************
1950  HOME : PRINT : HTAB 8: PRINT "PRESS RETURN IF A LIMIT
        IS CORRECT AS IS"
1960  PRINT : PRINT "CURRENT LOW SCALE LIMIT : ";LO
1970  PRINT "NEW LOW SCALE VALUE (MUST BE <"LB") ":
        INPUT T$: IF T$ = "" THEN 2000
1980  IF  STR$ ( VAL (T$)) <> T$ OR  VAL (T$) >  = LB
        THEN  VTAB  PEEK (37) - 1: CALL -868: GOTO 1970
1990 LO =  VAL (T$)
2000  PRINT : PRINT "CURRENT HIGH SCALE LIMIT : ";HI
2010  PRINT : PRINT "NEW HIGH SCALE VALUE (MUST BE >= "UB") ":
        INPUT T$: IF T$ = "" THEN 2040
2020  IF  STR$ ( VAL (T$)) <> T$ OR  VAL (T$) <UB THEN
        VTAB  PEEK (37) - 2: CALL -868: GOTO 2010
2030 HI =  VAL (T$)
2040  PRINT : PRINT "CURRENT SCALE STEP VALUE : ";INC
2050  INPUT "NEW SCALE STEP VALUE ? ";T$: IF T$ = "" THEN 2080
2060  IF  STR$ ( VAL (T$)) <> T$ OR T$ = "0" OR
        VAL (T$) > HI - LO THEN VTAB PEEK (37): CALL -868: GOTO 2050
2070 INC =  VAL (T$)
2080  HOME :DRWN = 0
2090  VTAB 10: HTAB 10: PRINT "(C) CONTINUE MODIFYING"
2100  PRINT : HTAB 10: PRINT "(M) RETURN TO MAIN MENU"
2110  VTAB 24: HTAB 17: PRINT "WHICH ? ";: GET A$:
        IF  ASC (A$) <31 THEN 2110
2120  IF A$ <> "C" AND A$ <> "M" THEN 2110
2130  IF A$ = "C" THEN 1760
2140  GOTO 3520

2150  REM *********************
2160  REM *   MODIFY LABELS   *
2170  REM *********************
2180  HOME : PRINT : HTAB 8: PRINT "PRESS RETURN IF A LABEL
        IS CORRECT AS IS"
2190  PRINT : PRINT "CURRENT VERTICAL LABEL : ";V$
2200  INPUT "ENTER NEW LABEL : ";T$
2210  IF T$ = "" THEN 2230
2220 V$ =  LEFT$ (T$,16)
2230  PRINT : PRINT "CURRENT HORIZONTAL LABEL : ";H$
2240  INPUT "ENTER NEW LABEL : ";T$
2250  IF T$ = "" THEN 2270
2260 H$ =  LEFT$ (T$,30)
2270  PRINT : PRINT "CURRENT GRAPH TITLE : ";TITLE$
2280  INPUT "ENTER NEW TITLE : ";T$
2290  IF T$ = "" THEN 2310
2300 TITLE$ =  LEFT$ (T$,30)
2310  IF P = 1 THEN 2400
2320  PRINT : PRINT "CURRENT 1ST Y VALUE LABEL : ";Y1$
2330  INPUT "ENTER NEW 1ST Y VALUE LEGEND LABEL : ";T$
2340  IF T$ = "" THEN 2360
2350 Y1$ =  LEFT$ (T$,4)
2360  PRINT : PRINT "CURRENT 2ND Y VALUED LABEL : ";Y2$
2370  INPUT "ENTER NEW 2ND Y VALUE LEGEND LABEL : ";T$
2380  IF T$ = "" THEN 2400
2390 Y2$ =  LEFT$ (T$,4)
2400  GOTO 2080

2410  REM ********************
2420  REM * MODIFY VERT VALUES*
2430  REM ********************
2440  HOME : VTAB 7: INPUT "MODIFY VALUES FOR WHICH HORIZONTAL
        ENTRY ? ";T$
2450  IF T$ = "" THEN 3520
2460  IF  VAL (T$) <1 OR  VAL (T$) > SIZE THEN 2440
2470 I =  VAL (T$)
2480  HOME : PRINT : HTAB 8: PRINT "PRESS RETURN IF A VALUE
        IS CORRECT AS IS"
2490  POKE 34,6: VTAB 10
2500 Q$ = "1ST": IF P = 1 THEN Q$ = ""
2510  FOR J = 1 TO P
2520  PRINT "PRESENT "Q$" VALUE OF ENTRY # ";I;" : ";Y(J,I)
2530  INPUT "ENTER NEW VALUE            : ";T$: IF T$ = "" THEN 2570
2540  IF ( STR$ ( VAL (T$)) <> T$) OR  LEN (T$) > 5
        THEN  VTAB  PEEK (37): GOTO 2530
2550 Y(J,I) =  VAL (T$): IF Y(J,I) > UB THEN UB = Y(J,I)
2560  IF Y(J,I) <LB THEN LB = Y(J,I)
2570 Q$ = "2ND": PRINT : NEXT : PRINT
2580  POKE 34,2: GOSUB 450: GOTO 2080

2590  REM *********************
2600  REM * MODIFY HOR. VALUES *
2610  REM *********************
2620  HOME : VTAB 7: INPUT "MODIFY VALUES FOR WHICH HORIZONTAL
        ENTRY ? ";T$
2630  IF T$ = "" THEN 3520
2640  IF  VAL (T$) <1 OR  VAL (T$) > SIZE THEN 2620
2650 I =  VAL (T$)
2660  HOME : PRINT : HTAB 8: PRINT "PRESS RETURN IF A VALUE
        IS CORRECT AS IS"
2670  POKE 34,6: VTAB 10
2680  PRINT : PRINT "PRESENT X-LABEL FOR ENTRY # ";I;" : ";X$(I)
2690  INPUT "ENTER NEW X-LABEL                  : ";T$
2700  IF T$ = "" THEN 2720
2710 X$(I) =  LEFT$ (T$,3)
2720  POKE 34,2: GOTO 2080

2730  REM ********************
2740  REM *   LOAD VALUES    *
2750  REM ********************
2760  HOME : HTAB 14: PRINT "LOAD VALUES"
2770  VTAB 8: INPUT "ENTER VALUES FILENAME : ";NAME$:
        IF NAME$ = "" THEN 3490
2780 SIZE = 0
2790  PRINT D$"OPEN"NAME$".DATA"
2800  PRINT D$"READ"NAME$".DATA"
2810  INPUT SIZE: INPUT P: INPUT LB: INPUT UB: INPUT LO:
        INPUT HI: INPUT INC: INPUT V$: INPUT H$:
        INPUT TITLE$: INPUT Y1$: INPUT Y2$
2820  FOR I = 1 TO SIZE: INPUT X$(I): NEXT
2830  FOR J = 1 TO P: FOR I = 1 TO SIZE: INPUT Y(J,I): NEXT : NEXT
2840  PRINT D$"CLOSE"NAME$".DATA"
2850  VTAB 11: PRINT "VALUES HAVE BEEN READ FROM FILE : ":
        PRINT : PRINT NAME$:DRAWN = 0: GOTO 3480

2860  REM *********************
2870  REM *    SAVE GRAPH     *
2880  REM *********************
2890  TEXT : HOME
2900  IF SIZE = 0 THEN  PRINT  CHR$ (7); CHR$ (7):
        VTAB 12: HTAB 5: PRINT "YOU HAVE NOT ENTERED
        A GRAPH !": GOTO 3480
2910  HTAB 15: PRINT "SAVE GRAPH": POKE 34,3
2920  VTAB 10: HTAB 9: PRINT "(H) SAVE HI-RES SCREEN": PRINT
2930  HTAB 9: PRINT "(V) SAVE VALUES"
2940  VTAB 24: HTAB 15: PRINT "WHICH ? ";: GET A$:
        IF  ASC (A$) <31 THEN 3520: PRINT A$;
2950  IF A$ <> "H" AND A$ <> "V" THEN 2940
2960  ONERR  GOTO 3170
2970  IF A$ = "H" THEN 3080
```

(Continued on next page)

Listing 1 *(continued)*

```
2980  HOME : PRINT : PRINT : INPUT "ENTER NAME FOR VALUES FILE : ";
        NAME$: IF NAME$ = "" THEN 3490
2990  PRINT D$"OPEN"NAME$".DATA"
3000  PRINT D$"DELETE"NAME$".DATA"
3010  PRINT D$"OPEN"NAME$".DATA"
3020  PRINT D$"WRITE"NAME$".DATA"
3030  PRINT SIZE: PRINT P: PRINT LB: PRINT UB: PRINT LO:
        PRINT HI: PRINT INC: PRINT V$: PRINT H$:
        PRINT TITLE$: PRINT Y1$: PRINT Y2$
3040  FOR I = 1 TO SIZE: PRINT X$(I): NEXT
3050  FOR J = 1 TO P: FOR I = 1 TO SIZE: PRINT Y(J,I): NEXT : NEXT
3060  PRINT D$"CLOSE"NAME$".DATA"
3070  VTAB 10: PRINT "VALUES HAVE BEEN SAVED IN FILE :":
        PRINT : PRINT : PRINT NAME$: GOTO3480
3080  IF DRWN = 0 THEN  HOME : PRINT  CHR$ (7); CHR$ (7):
        VTAB 12: HTAB 3: PRINT
        "YOU MUST FIRST RE-DRAW THE GRAPH !": GOTO 3480
3090  HOME : PRINT : PRINT : PRINT : INPUT "ENTER GRAPH NAME : ";NAME$
3100  IF NAME$ = "" THEN 3490
3110  PRINT D$"BSAVE"NAME$".PIC,A$4000,L$2000"
3120  PRINT : PRINT "GRAPH SAVED UNDER FILENAME :":
        PRINT : PRINT NAME$".PIC"
3130  GOTO 3480
3140  REM *********************
3150  REM * DISK ERROR ROUTINE*
3160  REM *********************
3170  TEXT : HOME
3180 ER =  PEEK (222):LN =  PEEK (218) +  PEEK (219) * 256
3190  IF ER = 255 THEN 3520
3200  PRINT  CHR$ (7); CHR$ (7)
3210  ON ER GOTO 3220,3220,3220,3230,3240,3250,
        3220,3260,3270,3280,3310,3220,3220
3220  HOME : VTAB 12: PRINT "ERROR OCCURED IN LINE "LN: END
3230  VTAB 11: PRINT "THE DISK IS WRITE PROTECTED.
        REMOVE THE WRITE PROTECT TAB OR INSERT
        A DIFFERENT DISK.": GOTO 3480
3240  VTAB 11: PRINT "FILE IS NOT A VALID VALUES FILE OR
        FILE IS NOT ON THIS DISK. PLEASE TRY AGAIN.":
        PRINT D$"DELETE"NAME$".DATA": PRINT D$"CLOSE"NAME$".DATA":
        GOTO 3480
3250  VTAB 11: PRINT "FILE BAR.GRAPH.CHAR (HI-RES CHARACTER
        SET) IS NOT ON THIS DISK. PLEASE INSERT THE CORRECT DISK.": END
3260  VTAB 11: PRINT "I/O ERROR. IS THE DISK DRIVE EMPTY OR
        THE DOOR OPEN ? IF NOT TRY A DIFFERENT DISK.": GOT 3480
3270  VTAB 11: PRINT "THIS DISK IS FULL. TRY A DIFFERENT DISK.":
        PRINT D$"DELETE "NAME$".PIC": GOTO 3480
3280  VTAB 11: PRINT "FILE "NAME$" IS LOCKED.":
        PRINT "DO YOU WISH TO UNLOCK IT AND
        OVERWRITE IT? ";: GET A$: IF A$ < > "N" AND A$ < >
        "Y" THEN 3280
3290  PRINT A$: IF A$ = "Y" THEN  PRINT D$"UNLOCK"NAME$".DATA":
        HOME : GOTO 2990
3300  GOTO 3480
3310  VTAB 11: PRINT "ILLEGAL FILE NAME.  PLEASE CHECK YOUR
        FILE NAME AND TRY AGAIN.": GOTO 3480
3320  REM *******************
3330  REM *    PRINT GRAPH   *
3340  REM *******************
3350  HOME : IF SIZE = 0 THEN 1770
3360  IF DRWN = 0 THEN 3080
3370  HTAB 15: PRINT "PRINT GRAPH": POKE 34,3
3380  VTAB 12: HTAB 2: PRINT "PLEASE ENSURE THAT THE PRINTER IS ON."
3390  VTAB 24: HTAB 7: PRINT "PRESS ANY KEY TO CONTINUE ";:
        GET A$: PRINT
3400  REM *******************
3410  REM *PLACE YOUR PRINTER*
3420  REM *GRAPHICS CODE HERE*
3430  REM *******************
3440  PRINT D$"PR#1": REM  PRINTER SLOT#
3450  PRINT  CHR$ (27)">": REM   UNIDIRECTIONAL PRINTING
3460  PRINT  CHR$ (9)"G2"
3470  PRINT D$"PR#0": HOME
3480  VTAB 24: HTAB 7: PRINT "PRESS ANY KEY TO CONTINUE ";: GET A$
3490  REM *********************
3500  REM *     MAIN MENU     *
3510  REM *********************
3520  TEXT : HOME : NORMAL
3530  INVERSE : HTAB 10: VTAB 2: PRINT " BAR GRAPH GENERATOR ": NORMAL
3540  VTAB 7: HTAB 11: PRINT "(E)  ENTER VALUES": PRINT
3550  HTAB 11: PRINT "(L)  LOAD VALUES": PRINT
3560  HTAB 11: PRINT "(S)  SAVE GRAPH": PRINT
3570  HTAB 11: PRINT "(M)  MODIFY VALUES": PRINT
3580  HTAB 11: PRINT "(R)  REDRAW GRAPH": PRINT
3590  HTAB 11: PRINT "(P)  PRINT GRAPH": PRINT
3600  HTAB 11: PRINT "(Q)  QUIT PROGRAM"
3610  VTAB 24: HTAB 11: PRINT "ENTER YOUR CHOICE : ";:
        GET A$: IF  ASC (A$) < 31 THEN 3610
3620  PRINT A$;
3630  IF A$ < > "E" AND A$ < > "L" AND A$ < > "S" AND
        A$ < > "M" AND A$ < > "R" AND
        A$ < > "P" AND A$ < > "Q" THEN 3610
3640  IF A$ = "E" THEN 530
3650  IF A$ = "L" THEN 2760
3660  IF A$ = "M" THEN 1760
3670  IF A$ = "S" THEN 2890
3680  IF A$ = "R" THEN 1080
3690  IF A$ = "P" THEN 3350
3700  IF A$ = "Q" THEN  HOME : END
```

Listing 2

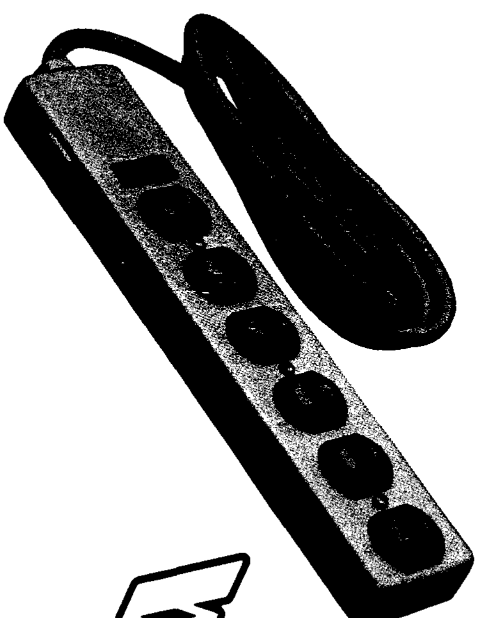
```
*6000.6374

6000- 42 00 88 00 8E 00 90 00 97 00 A2 00 B3 00 C3 00
6010- D0 00 DE 00 E8 00 F1 00 FA 00 0D 01 16 01 1B 01
6020- 21 01 24 01 2A 01 39 01 40 01 4C 01 59 01 64 01
6030- 72 01 7F 01 8E 01 9D 01 AA 01 B1 01 B9 01 C7 01
6040- D1 01 DF 01 E9 01 FB 01 07 02 13 02 1D 02 28 02
6050- 34 02 3D 02 4B 02 58 02 65 02 6D 02 7C 02 8C 02
6060- 9A 02 AD 02 B8 02 C1 02 D0 02 E0 02 EB 02 F3 02
6070- FD 02 0A 03 19 03 25 03 2F 03 3E 03 4A 03 55 03
6080- 60 03 6D 03 71 03 73 03 3E 24 2D 36 04 00 00 00
6090- 09 0C 18 24 24 04 00 C9 0B 18 08 18 08 18 30 4D
60A0- 24 00 21 24 24 6C 36 36 36 0C 58 1F 1F 0C 18 0D
60B0- 0D 04 00 A8 2D 0C 88 23 1C 3F 1C 64 8C 75 1F B6
60C0- 16 20 00 64 0C 0C 0C FC 3B 2E 96 52 29 3C 04 00
60D0- 29 0D 1C 0C 1F 1F 26 08 0D 18 E4 17 26 00 49 09
60E0- 18 08 18 08 18 B8 04 00 49 E1 0F 18 24 0C 0C 04
60F0- 00 0D 18 0D 18 24 1C 1C 04 00 09 24 24 24 5D 0A
6100- 17 D7 BA 0C 18 08 18 8C AA 11 15 04 00 09 20 24
6110- EC 93 6F 29 04 00 C9 0D 18 24 00 08 18 08 2D 04
6120- 00 09 04 00 64 0C 4C 63 24 00 29 65 24 24 1C 3F
6130- 17 36 36 0C 88 63 0C 04 00 29 E5 24 24 BC 04 00
6140- 2D 2D DC CB 0C 0C 0C E4 3F 17 04 00 A8 2D 0C 3C
6150- 38 67 49 23 1C 3F 17 04 00 49 24 24 24 17 1E 1E
6160- 2E 6D 04 00 A8 2D 0C 24 0F 18 DD AF 1B 24 2C 2D
6170- 25 00 29 65 3C 38 3F 36 0C 18 60 0C 8C 04 00 08
6180- 18 08 18 08 18 2D 2D D5 1E 1E 1E 36 04 00 29 65
6190- E4 3F D5 33 0C 18 20 0C 2D 94 11 24 00 29 0C 0D
61A0- 18 3C 3F 1C 64 2D 8F 31 04 00 09 08 18 0C 18 04
61B0- 00 C9 0D 58 23 0C 18 04 00 49 E1 0F 18 0F 18 0D
61C0- 18 0D 18 0D 18 04 00 08 18 2D 2D 0C D8 39 3F 27
61D0- 00 0D 18 0D 18 0D 18 0F 18 0F 18 0F 18 04 00 09
61E0- 0C 18 2C 28 E0 3F 17 04 00 29 2D DC 1B 24 24 0D
61F0- 18 0D 58 1A AD 36 FE 39 2C 04 00 24 24 2C 28 AD
6200- 36 36 26 18 3F 04 00 24 24 24 2D 75 F6 3F 96 2D
6210- 0C 24 00 29 65 DF 23 24 64 2D 15 04 00 24 24 24
6220- 2D 15 15 36 1E 1E 27 00 24 24 24 2D 2D 96 3B B7
6230- 2A 2D 04 00 24 24 24 2D 2D 96 3B 27 00 29 2D 2C
6240- FF 33 0C 18 24 0D 18 2D 15 04 00 24 24 24 4D 31
6250- 36 36 26 08 D8 3F 04 00 29 3D 20 24 20 08 98 92
6260- 0C 18 6F 04 00 20 95 2D 0C 24 24 24 00 24 24 24
6270- 4D E9 1A 1E 1E 8E 09 D8 15 15 04 00 24 24 A4 92
6280- 0A 18 08 18 08 18 96 92 29 2D 04 00 24 24 24 8C
6290- AA 0D 18 0D 18 36 36 36 04 00 20 16 0C 18 20 24
62A0- 4D 31 36 36 26 08 D8 E3 58 B2 09 04 00 29 65 24
62B0- 24 1C 3F 9D 33 36 26 00 24 24 24 2D 75 F6 3F 04
62C0- 00 29 0C 0C 24 3C 38 AF 1B 36 36 4D 18 56 21 00
62D0- 24 0C 98 24 2C 2D 15 D5 1E 3F 56 09 18 0E 26 0C
62E0- A8 2D 0C E4 3F 1C 2C 28 AD 04 00 09 24 24 24 3F
62F0- 4D 25 00 29 65 24 24 FC 1B 36 36 26 00 08 15 0D
6300- 18 0D 18 24 24 DF 33 36 26 00 24 24 24 4C 89 36
6310- 36 36 0F 18 0F 18 17 04 00 2C 28 28 28 20 DF 33
6320- 15 55 AA 26 00 09 24 2C 28 20 DF 33 0E 04 00 2D
6330- 2D DC 1B 0C 0D 18 0D 18 0D 18 3C 3F 27 00 29 2D
6340- DC 27 24 24 2D F5 33 36 04 00 49 21 1C 0F 18 0F
6350- 18 0F 18 24 0C 2D 25 24 24 3C 3F 4E 36 36 04 00
6360- 08 18 08 18 08 58 61 12 15 DF 63 04 00 2D 2D 05
6370- 00 05 00 7D 00
```

MICRO

another **FLEXIDUCT® OFFICE SAFETY PRODUCT**

When you least expect it, ZAP!!!



In a few millionths of a second, common electrical surges and spikes can enter your data processing equipment and cause memory loss, false logic and misregistration. Surges very often do permanent damage to microcircuitry.

FLEXIDUCT Surge Suppressors catch surges and spikes before they have a chance to enter your equipment. In billionths of a second (Nanoseconds),**FLEXIDUCT** Surge Suppressors dissipate surges and spikes from any side of the line (most protect only one side).

The multi-outlet unit is ideally suited to the computerized workstation. It gives you the advantage of circuit breaker protection with plenty of outlets for data processor and peripheral equipment.

No computer should be without the protection of a **FLEXIDUCT** Surge Suppressor...**especially yours!** Write or call for further information. Available from office products retailers.

FLEXIDUCT® Surge Suppressors

a product of Winders & Geist, Inc. P.O. Box 83088 Lincoln, NE 68501 402/474-3400

MICRO™ Apple Software Catalog

The Creator Software That Writes Its Own Programs

Software Technology for Computers, Inc. has announced the introduction of **THE CREATOR**, a new software package for personal computers designed to enable any user to tailor a variety of applications to their particular requirements — with virtually no programming knowledge.

With THE CREATOR loaded onto an Apple II or IIe, the user can design his own program in two ways. After selecting one of ten ready-to-customize work areas, he answers simple English-language questions, which appear in a logical step-by-step process on his screen. THE CREATOR does the programming and the custom application is completed and ready to use in as little as ten minutes.

For users who want to design unique individualized applications outside of the ten work areas, THE CREATOR offers many additional English-language routines, which provide the utmost in flexibility.

THE CREATOR is currently available through major software distributors throughout the country at the suggested retail price of **$250** for the Apple II or IIe. Contact Software Technology for Computers, Inc., 153 California St., Newton, MA.

Sir-Tech Software Introduces Third Wizardry Scenario With "Windo-Wizardry"

Sir-Tech Software, Inc. has unveiled its **WINDO-WIZARDRY** high-resolution display manager for the Apple II through its latest Wizardry recreational program called "Legacy of Llylgamyn."

WINDO-WIZARDRY allows programmers to create and manipulate rectangular windows, areas of any size, inside which text and graphics can be placed. Each window as several attributes that allow WINDO-WIZARDRY to generate and maintain an image on the screen. These include the window's size, position, and priority. "When windows are moved, created, removed, or changed in priority, WINDO-WIZARDRY automatically regenerates the hi-res display; and so, the contents of previously obscured windows may become visible. WINDO-WIZARDRY puts more information at the fingertips of the user by careful use of overlapping windows.

For more information contact Roberts Advertising Agency, Inc., P.O. Box 686, 7593 Morgan Office Campus, Liverpool, NY 13088; (315)451-4434.

MICRO™

MICRO™

Apple Hardware Catalog

Sup'r Terminal
for the Apple II

Expand your Apple II's terminal features with the best engineered 80-character upper/lower-case plug-in board. **Sup'r Terminal** is forward-engineered for past, present, and future compatibility. It operates synchronously with the Apple II for faster clearing, faster scrolling, and "clean screen" operation. Sup'r Terminal is engineered for the best display, for compatibility, for the user, and for performance.

M & R ENTERPRISES
910 George Street
Santa Clara, CA 95050
(408)980-0160

The Apple Blooms!
for the Apple II and Apple IIe

Hollywood Hardware's new program development package installs such crucial professional features as a Global Program Line Editor, definable Function Keys, Output Formatting, and Structured Program Aids with one special firmware card. The package requires no disk loading and uses *no* memory space, yet it speeds editing up to five times using Insert, Delete, Search, Replace, and more. Powerful macros perform common tasks like Catalog and List with one keystroke, and you can program and nest custom macros to produce complex command sequences.

Additional "&" utilities extend Applesoft (e.g., "IF, THEN, ELSE" and "PRINT USING"), perform Searches, Number Base Conversions, Garbaged Program Recovery, and others.

Future utilities from Hollywood Hardware (including: Renumber, Append, Disk Diagnostics, etc.), and user programs are supported with a documented memory manager and six open sockets totalling 24K of ROM expansion capabilities.

This product is supplied with a 75-page manual, quick reference guide, demon-disk, and 30-day trail period.
$190.00

Hollywood Hardware
6842 Valjean Avenue
Van Nuys, CA 91406;
(213)989-1204

Keywiz VIP (Very Intelligent Peripheral)
for the Apple II+ and Apple IIe

Keywiz VIP is the first user programmable keyboard no software interaction. Standard features include 62 user programmable keys (31 lower case/31 shifted) per keyboard, storage of up to 4 (62-key) keyboards in its own memory, and the ability to switch between keyboards at the touch of a button making all 248 user-defined keys available. You can reprogram any key at anytime.

Creative Computer
1044 Lacey Rd.
Forked River, NJ 08731

MICRO

MICRO™ Apple Reviews

Product Name: **The Visible Computer: 6502**
Equip. req'd: Apple II with Applesoft, 48K RAM, and DOS 3.3
Price: $49.95
Manufacturer: Software Masters
3330 Hillcroft/Suite BB
Houston, TX 77057

Description: This "machine-language teaching system" consists of a 142-page book plus a disk containing a 6502 simulator and sample programs. As the text teaches machine-language programming, the simulator makes visible (in hi-res graphics) what goes on inside the 6502.

Pluses: The text uses many helpful analogies that gives a clear conceptual picture of what happens inside the CPU; it even conveys a sense of adventure in learning machine-language programming. The simulator is an excellent learning tool, helping the learner retain concepts rather than simple facts. And it's fun to watch!

Minuses: It's hardly worth mentioning, but punctuation and spelling could be more accurate.

Documentation: The text and simulator program fit together very nicely to help the learner.

Skill level required: Some familiarity with Applesoft BASIC (the more the better); no familiarity with machine language.

Reviewer: Jon R. Voskuil

Product Name: **Modula-2**
Equip. req'd: Apple II, Apple II+, Apple II/e with 64K, two disk drives and the Apple version of the UCSD Pascal System
Price: $495.00
Manufacturer: Volition Systems
Box 1236
Del Mar, CA 92014
(714) 481-2286
Copy Protection None
Language: Compiler written in Pascal, remainder of the system written in Modula-2

Description: *Modula-2* is the third language to be developed by Niklaus Wirth, the designer of Pascal. *Modula-2* contains much of the language philosophy found in Pascal yet expands the capabilities into areas required by a systems language. Pascal was originally designed only as a teaching aid.

Pluses: Experienced Pascal programmers are familiar with many quirks and weaknesses of the language. *Modula-2* begins where Pascal stopped. Numerous functions of the language are implemented at the primitive level allowing the user greater power. For example, Pascal contains only a few, very simple, Read/Write commands, which, under type error, will cause the system to hang (e.g., read a character when a number was expected). *Modula-2* has many variations of I/O commands including ReadInt, WriteInt, WriteString, WriteHex, ReadWord, WriteWord, etc. Perhaps the most significant addition, however, is the use of modules.

Minuses: Although *Modula-2* operates under Apple's version of the UCSD system, Pascal source code is not compatible with it. Thus, a fair amount of translation will be needed to rewrite existing Pascal source to convert it to Modula-2. Generally, compiled Pascal code can be expected to execute directly. However, there are two p-codes (IDSEARCH & TREESEARCH) that have been removed by Volition Systems and could cause some incompatibility when executing previously compiled programs.

Skill level required: *Modula-2* will be of interest primarily to experienced Pascal programmers.

Reviewer: David Morganstein

MICRO™

REBOOTING DOS

by Ernest P. Gagnon
and Kenneth Gagnon

Occasionally one discovers (usually after entering a somewhat lengthy Applesoft program from the keyboard) that he or she has neglected to load DOS into memory. Additionally, DOS has been known to mysteriously disappear if the RESET key is used to recover from a hing machine-language subroutine or if the wrong memory location is POKED. Once that happens all subsequent attempts to communicate with the disk drive in order to SAVE, LOAD, or RUN a program will be unsuccessful. Loading DOS by typing 'PR#6' to boot a disk will destry the program in memory since the DOS booting process uses portions of memory in which the program (particularly Applesoft) and its pointers are stored.

If the program has been previously saved to disk, then recovery involves the simple process of booting the disk and reloading the program. But what can you do if the program has not previously been saved? Fortunately, there is a relatively simple technique for gracefully recovering from this otherwise frustrating situation without loasing the program you spent so much time entering. The technique described here involves moving the Applesoft program from its normal place of residence in memory to a temporary location so that DOS can be loaded without 'stepping' on your program.

While the same technique can be used to 'hide' machine language and other types of programs during the DOS boot, we will specifically limit the discussion in this article to Applesoft BASIC programs. Prior to describing the technique, we will look at memory allocation for Applesoft programs and examine the DOS loading process.

Applesoft Memory Allocation

Applesoft programs reside in RAM memory starting at location $0800 and building toward higher memory locations. As an example, type in the listing for 'Test Program' provided in listing 1. RUN it to establish how the program works and enter the Monitor by typing 'CALL −151' and pressing Return. The asterisk (*) prompt lets you know that you are in the Monitor mode. Type '0800.085F' followed by a Return; this directs the computer to display the contents of memory from location $0800 through $085F. Address $085F was somewhat arbitrarily chosen simply to display the first 60 bytes of the program. Location $0800 usually contains the number 00 for Applesoft programs. The next two locations designate the start of the first statement line of the program; each program line begins with a pointer to the succeeding line of the program.

For 'Test Program' the contents of locations $0801 and $0802 contain the values 18 and 08, respectively. These values point to the starting address of the second statement number of the program at location $0818 (the low byte of the address always preceeds the high byte as stored in memory). Locations $0803 and $0804 contain the values 0A and 00, respectively, which represent (in ASCII code) the beginning line number of the Applesoft program. The decimal equivalent for $000A is 10; the program listing provided in listing 1 shows that the starting line number is indeed 10.

The next several bytes of memory include the information contained in the Applesoft line, comprising ASCII code characters and the so-called tokens for Applesoft's reserved words such as PRINT, GOSUB, and END. It is beyond the scope of this brief article to go into depth regarding the interpretation of Applesoft programs in machine language.

We can examine the contents of memory preceding the end of the program at $08D3 by typing, in the Monitor mode, "08C0.08D3" and pressing Return. By carefully examining the contents of the memory locations displayed in the listing we can determine the contents of the ending line of the program. The last statement line of "Test Program" starts at location $08CC. Reading forward from that memory location we find the values 96 and 00; i.e., the number $0096, which converts to 150 in decimal. This corresponds to the last statement number in the program listing of figure 1.

In order to preserve this program in memory while booting DOS 3.3, it is necessary to relocate the entire program from starting address $0800 through the end of the program to a safe hiding place in memory.

The DOS Booting Process

DOS boots in several stages loading first into low RAM address locations such as $0800 and finally settling down in the highest reaches of RAM. During this bootstrap process DOS effectively "steps" on large chunks of territory in the very memory locations where the Applesoft program is stored. An excellent description of the booting process is provided in Chapter 5 of reference 4. There is a vast expanse, however, extending from memory location $5000 (20480 decimal) to beyond $9000 (36864 decimal), which is untouched during the booting process thereby providing an area in which we can relocate our Applesoft program during the booting process.

There is one final bit of information we should understand before getting on with loading DOS. If we were to simply relocate our Applesoft program to the designated safe portion of memory, boot DOS, and restore the program to its original site in memory, we would find that the program would neither RUN nor SAVE to disk properly. The problem stems from the fact that the computer uses certain designated memory locations on Page 0 ($0000 through $00FF) as pointers to the start

(Continued on next page)

and length of the program. The DOS booting process writes over several of these locations also. Consequently, we must preserve a portion of the Applesoft pointers on Page 0 in addition to our program during the booting process in order to survive unscathed.

Program Save and DOS Boot Procedure

The procedure, then, for preserving an Applesoft program during a DOS boot is as follows:

1. Enter the Monitor mode by typing "Call-151". The prompt will be a *.
2. Determine the ending address of the Applesoft program by typing "AF.B0" followed by Return. Record the hexadecimal bytes following the AF and the B0. Assume these are LL (low byte) and HH (high byte), respectively. Therefore, the ending address of the program would be $08D3. Determine the length of the program by subtracting the starting address from the ending address and adding 0001 to the result. For example, $08D3 − $0800 + $0001 = $00D4. The result, $00D4, is the total length of "Test Program". Note: All of the arithmetic operations must be performed in hexadecimal to arrive at the correct result.

An example will illustrate the technique employing the values of the current example. Start by entering the high byte of the ending address and subtracting the high byte of the starting address; that is, enter "08-08" followed by Return. The result will be 00. Next enter the low byte of the ending address and subtract the low byte of the starting address; that is, enter "D3-00" followed by Return. The result is obviously D3. To complete the operation, reenter the low-order byte and add 1, D3 + 1, which results in D4. Combining the high-and low-order bytes in the proper order yields $00D4, which, for this example, is the length of the program.

3. Select a starting address to which the program will be moved during the DOS booting process. The starting address, SSSS, must not be less then $5000 and must be greater than the ending address of the program at HHLL. A starting address of $5000 is recommended, if suitable.
4. Move the program to its temporary location for the booting process by typing "SSSS <0800.HHLLM" and pressing Return as in the following example for "Test Program":

5000 < 0800.08D3M

Caution is urged when entering the source and destination addresses used in the block-move commands as it will be difficult, if not impossible, to recover from incorrectly typed addresses. Verify each entry prior to hitting Return.

5. Preserve the necessary pointer from Page 1 by relocating them in memory as follows:

9060 < 60.FFM

This assumes, of course, that the program is not so long as to have written beyond location $9000.

6. Load DOS 3.3 by inserting a suitable disk into drive 1 and, from the Monitor mode, type 6 CTRL P followed by Return. The disk drive will start up and DOS will be loaded into memory.
7. Return to the monitor mode by typing "Call-151". Restore the Page 0 pointers by typing

60 <9060.90FFM

8. Restore the Applesoft program to its original location in memory by typing "0800 <SSSS.EEEEM," inserting appropriate values for SSSS and for EEEE. The value of the starting address SSSS must be the one selected in Step 3 and used in Step 4 to relocate the program. The value of the ending address EEEE is determined by adding the starting address SSSS to the length of the original Applesoft program established in Step 2. Using the values for "Test Program":

SSSS = $5000
Length = $00D4
EEEE = $50D4

Consequently, the move command would be

0800 < 5000.50D4M

9. Leave the Monitor mode and reenter the Applesoft BASIC-language mode by typing "CTRL C" followed by Return.

You are now able to LIST, RUN, and SAVE the program in the usual function. DOS has been loaded into the machine while preserving the resident Applesoft program. RUN the program to verify that the procedure has been executed correctly. File this article in an accessible place as it could come in handy someday when DOS mysteriously disappears.

Listing 1

```
10  REM  * TEST PROGRAM *
20  HOME
30  A$ = "TEST":B$ = "PROGRAM":X1 =
    3:X2 = 32: GOSUB 100
40  A$ = "*":B$ = "*":X1 = 1:X2 = 40:
    GOSUB 100
50  VTAB (23)
60  END
100 FOR I = 1 TO 23
110 HTAB (I + X1): VTAB (I): PRINT A$
120 HTAB (X2 - I): VTAB (24 - I):PRINT B$
130 FOR K = 1 TO 30: NEXT K
140 NEXT I
150 RETURN
```

References

1. Espinosa, C.: *Apple II Reference Manual*, Apple Computer Co., Cupertino, CA, 1979.
2. Osborn, T.: "Apple Slices," MICRO - The 6502/6809 Journal, No. 54, November 1982, pp. 86-87.
3. Woodward, K. G.: "Apple Byte Table," MICRO - The 6502/6809 Journal, No. 41, October 1981, pp. 88-92.
4. Worth, D. and P. Lechner: *Beneath Apple DOS*, Quality Software, 1981.

Kenneth Gagnon is a student at Manchester High School; Ernest is a Research Engineer at United Technologies Research Center. The authors may be contacted at 151 Ralph Road, Manchester, CT 06040.

MICRO™ Apple Slices

by Jules H. Gilder

Three new software packages have recently been introduced that are both useful and interesting. They are as follows: an enhanced version of Applesoft, a disk drive analyzer program and a new version of the popular spelling checker, Sensible Speller.

The most interesting new piece of software we have seen recently is Beagle BASIC, which is a utility program that allows you to load Applesoft into a language card (even if you have Applesoft ROMs in your computer) and lets you enhance its capabilities. Would you like to have the ability to write IF...THEN...ELSE statements? Would you like to be able to swap the contents of two variables without the need for an intermediate third variable (comes in handy when doing sorts)? How about using named subroutines — say GOTO or GOSUB PRINTOUT rather than going to the printing routine by saying GOTO or GOSUB 49671. All of these things and more are now possible with Beagle BASIC.

Another feature of Beagle BASIC is the ability to rename the existing Applesoft commands. Since Applesoft is in RAM, this is easy to do. Thus, if you wanted to, you could change HOME to CLS or shorten INVERSE to INV. Beagle BASIC also lets you change the error messages. This could be helpful to make programs more user friendly.

In addition to all this, Beagle BASIC allows you to replace many Applesoft commands that are not frequently used (e.g. STORE, RECALL, cassette SAVE and LOAD, etc.) with newly defined commands such as TONE, where you give the pitch and relative length of duration, SWAP, which allows you to swap any two variables, HSCRN, which tells you the color of any hires coordinate and many more. The program is very well designed and extremely easy to use. It comes on a standard. unprotected DOS 3.3 diskette and only requires that you have a language (or other RAM card) in slot zero and a copy of Applesoft either on disk or in ROM.

Checkout Your Drives

If you use your Apple a lot, you may have encountered a situation where your disk drive started to act funny, or not work at all and didn't know what to do. Because disk drives are mechanical devices, they are more prone to failure than other parts of your Apple computer. Now, with a new diagnostic diskette from Verbatim, it is possible for even the most non-technical person to check out his or her disk drives regularly and avoid these problems.

Known as the Datalife Disk Drive Analyzer, this program sells for $69.95 and can perform four comprehensive tests on any of your floppy disk drives in less than two minutes. The tests check the radial alignment of the disk drive head to make sure that it is properly centered over the diskette tracks, the speed at which the the diskette rotates, the mechanical clamping mechanism that holds the diskette in place and makes it rotate, and, whether or not your disk drive reads and writes data accurately.

Unlike other disk diagnostic programs that have been available in the past, this one does not require additional scratch diskettes or any technical knowledge. With a total checkout time of two minutes, it is possible for even non-technical users to test their disk drives once a week and know right away if they have any problems with their drives.

An Apple Spelling Checker

While there are several spelling checkers available that run on the Apple computer, one of them outshines them all: the Sensible Speller from Sensible Software. Early versions of this program have always been popular, but have suffered from one or two deficiencies. The recently introduced Version IV corrects all of them. Unlike other spelling checkers, this one has a very wide range of compatibility.

While most programs will work with only one operating system, this one works with several. It can of course work with Apple's current operating system, DOS 3.3, but it can also be used with the older DOS 3.2. Beyond that, if the user desires, he can also use Sensible Speller on files generated by Apple's Pascal system or even on text files generated by the CP/M operating system. Finally, it also works on files that are generated by two word processors that use their own protected DOS.

Another nice feature of the program is that it works both on standard text files and on binary files that contain text (such as those generated by PIE Writer and Magic Window), and it adjusts to the file type used automatically. As far as compatibility with word processors is concerned, Sensible Software indicates that it is compatible with at least 25 of them. I tested the program on word processors not included on Sensible's list, and it worked perfectly.

The dictionary for the speller comes on two diskettes, the first containing about 44,000 words and the second about 37,000 words. The 44,000 words on the main dictionary diskette were chosen wisely, because, in repeated use, only a handful of the remaining unknown words are found on the second dictionary diskette. The dictionaries are stored in a compressed format, and the user can build his own dictionary either from words that he has corrected, or by reading a separate word file. This latest version of the program allows you to interactively correct text while earlier versions only let you mark the suspect word and forced you to go back to your word processor to do correction. In addition, when corrections are being made, the program displays the suspect word in context so that it is easier for the user to determine just what the word should be.

The one major flaw in the Sensible Speller occurs in its dictionary maintenance operations. If you want to add new word to an existing dictionary, you must always have an extra blank disk around, because instead of just adding on to the existing file, a whole new dictionary disk must be created.

The program is fast and easy to use and can be mastered in less than an hour. In addition, the manual takes you step-by-step through a sample session and explains what is happening at each stage. The program costs $125 and comes on a copy protected disk.

MICRO™

Interface Clinic

by Ralph Tenny

Last month's interface experiment was based on a Universal Asynchronous Receiver/Transmitter (UART). Before we set up to program this interface, we must tie up a couple of loose ends. First, many 6809 microprocessor systems use a related IC called the Asynchronous Communications Interface Adapter (ACIA), and we should examine the differences between the UART and the ACIA. The most obvious difference is that the ACIA comes in a 24-pin package instead of the 40-pin package of the UART, with no reduction in capability. In fact, the ACIA is programmable and has greater versatility. When the ACIA was introduced, it also was the only one-power supply UART available (the two devices specified last month are reasonably new; UARTs used to require a +12-volt supply also).

What do we give up to lose 16 package pins? An ACIA does not have separate parallel output and input bus lines; instead, there are eight data I/O pins that communicate directly with the microprocessor. Therefore the ACIA could not have been used in last month's project, but it has a few advantages. All status signals (error flags, data ready, etc.) are kept in a status register read by the processor over the data bus. The clock input sets the baud rate, just as with the UART, but an internal control register can be programmed for either /16 or /64 operation to allow software selection of two asynchronous baud rates (1200 baud and 4800 baud, for example), and a synchronous /1 mode can be selected for special applications. Also, the serial attributes — number of data bits, number of stop bits and parity choices — are programmable.

I had to do some *worst case design* while designing the interface adapter. This came up when I tried to use the RESET line into U2 (pin 2) as a UART clock. The Intersil IM6402 UART worked perfectly with this clock source, but the Radio Shack part wouldn't. I finally traced this down to the specified clock pulse width. The IM6402 is specified for .4 μSec minimum pulse width, which the RESET pulse meets; the Radio Shack part requires 1 μSec pulse width minimum. I had to clock the UART from the counter MSB output to meet clock minimums. I won't admit how much time this oversight cost me, but it could happen to you!

Back to the interface adapter. Let's review how it works (see figure 1 for a reminder of the circuit). The 9600 Hz clock input is internally divided by 16 to allow repeated sampling of serial input waveforms. New parallel output data appears on the RD lines each time the computer sends a new binary character, and remains latched until new data is sent. A slow clock on pin 23 of the UART initiates a data transmission consisting of the eight data bits sampled by the eight DB lines. This clock signal is sent to the CD input to be used for interrupt programming.

Simple BASIC statements can be used to output data to the UART, but those of you with Color Computers will not be able to input data without a special program we will develop. If you use this direct statement, the UART will receive a data value: PRINT#-2, CHR$(X); (X will be a decimal number between 0 and 255). Note that without the semicolon, the data will be followed by a carriage return! Here's a gotcha: if you have an early model CoCo (1.0 BASIC ROM), RD7 will always be high unless you have the software patch that allows 8-bit serial output. Certain versions of other computers may also have a problem sending 8-bit binary serial data, so you need to check this out.

On input, you must sample the CD signal somehow to know when a new serial transmission begins. Alternatively, you can put the computer in a loop, sampling the SERIAL IN line while watching for a start bit. If you have a computer that does not use a bit-banger serial port (such as an Apple with a serial card), the UART or ACIA on this card will handle the serial input/output automatically if you set the baud rate to 600.

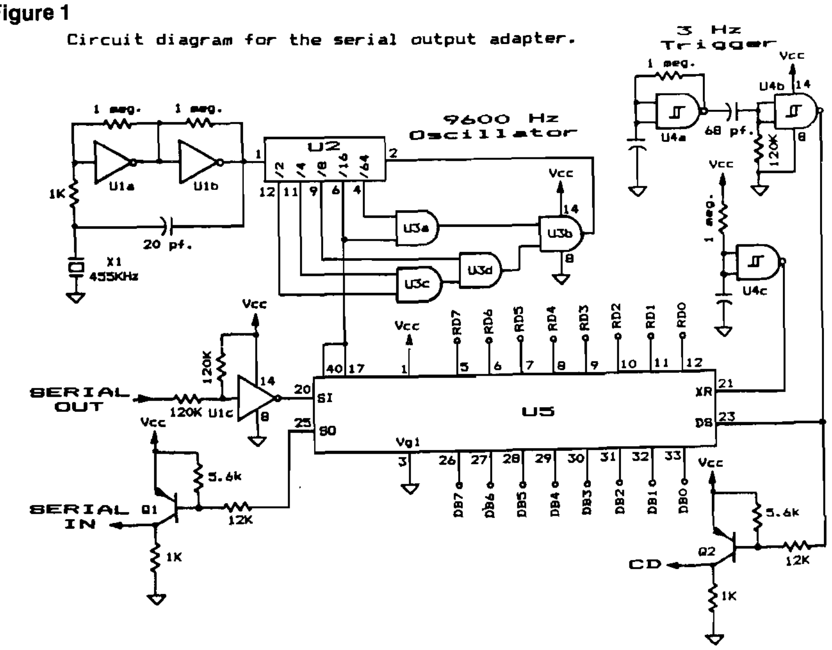
If you have a bit-banger serial port, you have two programs to develop: the first must help you to find a time constant value to drive the port. This program will also be an output program. The second program will be an input program that uses a similar software timing loop. The transmit program can be built around the subroutine shown in figure 2. This routine is entered with the 8-bit data value in the A accumulator, and accumulator B is used as a counter for the eight bits to be sent.

My calibration/transmit program is shown in listing 1, and it has several features to note. The DELAY

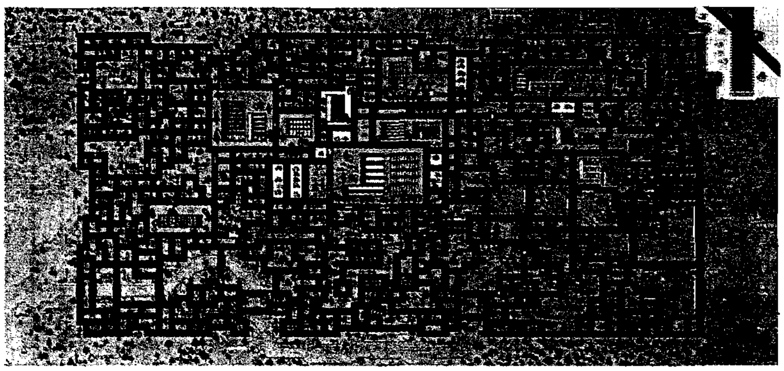
(Continued on page 108)

Figure 1

Circuit diagram for the serial output adapter.



You've got the loot ... now, *GETAWAY!* to a great new game from the Atari® Program Exchange!

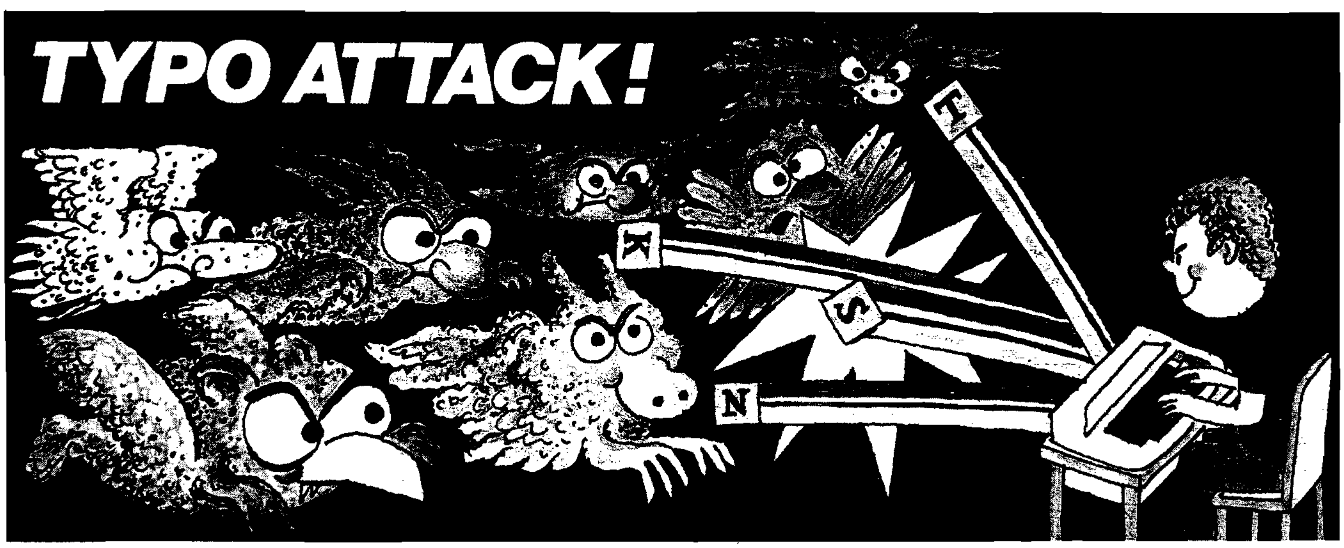


It's *all* there! The squall of sirens, the crazy turns down endless city streets, the anxious search for ill-gotten gain, the race against time for a safe place to stash your cash! Now your gas tank is nearly empty and night is about to fall. The coppers are closing in fast. Before you learn again that crime doesn't pay...Quick! ***GETAWAY!***

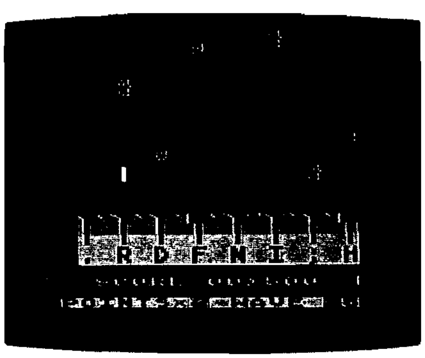
Ask for *GETAWAY!* at your local Atari software retailer, or order direct. Phone 800-538-1862, or 800-672-1850 in California. Or write Atari Program Exchange, P.O. Box 3705, Santa Clara, CA 95055.

Cassette (410)	APX-10195 32K	$29.95
Diskette (810)	APX-20195 32K	$29.95

For direct orders, add $2.50.



You're in for a nasty spell ... unless you stop the Typos!



In the dusky world beneath your keyboard the gruesome Typos dwell ...waiting to attack! Term paper due tomorrow? Got to get that book report typed? Fool! The Typos will devour your letters as you type! That could spell D-O-O-M-E for *you!*

Before you start typing, get down to the real work: destroy the Typos before they destroy your proase... uh, proze...prrrose...Oh NO! THE TYPOS!!! Get ***TYPO ATTACK***, a grand and glorious game from Atari® Program Exchange. It might even improve your typing!

Ask for ***TYPO ATTACK*** at your local Atari software retailer, or order direct. Phone 800-538-1862, or 800-672-1850 in California. Or write Atari Program Exchange, P.O. Box 3705, Santa Clara, CA 95055.

Cassette (410):	APX-10180	8K	$29.95
Diskette (810):	APX-20180	16K	$29.95

For direct orders, add $2.50.

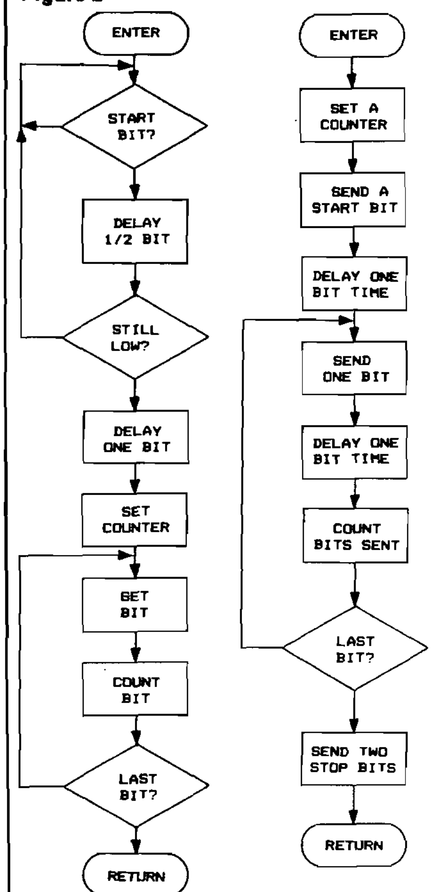
subroutine at the end of the program is a general-purpose routine; enter it with a 16-bit number in the X register. The MARK and SPACE subroutines handle the start and stop bits, and the MAIN PROGRAM is written with the idea of sending known patterns to a printer. This program generates the binary characters 01010101 and 10101010, which were chosen because they have alternating bit patterns. My serial/parallel printer interface is set for seven bits, so the printer prints "U" and "*". If you send an 8-bit pattern, the "*" will be replaced by a graphics character if your printer can handle it. A brief delay separates the two characters so you have a chance to stop transmission by pressing any key on the keyboard (the JSR POLCAT instruction). A carriage return is then sent and the loop repeats until you press a key.

After the program is assembled and entered into memory, run it using a debug monitor program such as ORACLE (Spectral Associates) or HUMBUG (Star Kits). Set a breakpoint for the RTS at the end of the MAIN PROGRAM segment, and set the value in location 149/150 ($95/$96) to $AB for your first attempts. Run the program with the printer attached instead of the serial adapter, and see if the printer prints "U*" on successive lines. If not, adjust the $AB value up or down until these characters are printed. Once you find a value that works, continue to vary it to find the limits of the value that works. My setup ran with delay values from $9D to $B8, and the midpoint of this range is $AB. Once this is working, MAIN PROGRAM can be modified to send selected single characters to the serial adapter.

Listing 2 shows a polling routine, which inputs data from the serial adapter. You can use an eight-position dip switch to set various values for the adapter to send, but I used with the binary value 01010101 ($55) as a test value. This listing uses the same DELAY routine as listing 1, and has provision for saving each sample of the input in a series of buffers to allow you to check the time constant range as was done in listing 1. After each input run, display the buffer area to see what was input by the LDA PORTIN statement at $3041. When the program and adapter are working correctly, eight successive memory locations should hold alternating values of 05 and 04; location $305E should contain $55.

Next month we will do interrupt programming on the serial interface adapter and begin looking at real-world interfaces that allow us to use the serial adapter for controlling machinery, lights, etc.

Figure 2



Flow charts for RECEIVE (left) and TRANSMIT

Listing 1

Calibrate Serial Adapter and Transmit Data.

```
                         * THIS PROGRAM WILL CALIBRATE A DELAY LOOP
                         * TO SEND AND RECEIVE 600 BAUD SERIAL DATA
                         * OVER THE COLOR COMPUTER SERIAL PORT.
                         * EQUATES
                 0095    BAUD     EQU    149        LOCATION OF DELAY CONSTANT
                 A000    POLCAT   EQU    $A000      KEYBOARD SCAN
                 FF20    SEROUT   EQU    $FF20      RS232 OUT (BIT 1)
                         * MAIN PROGRAM
3000                              ORG    $3000
3000 86   55             START    LDA    #$55       "U"
3002 1A   50                      ORCC   #$50       KILL INTERRUPTS
3004 8D   13             SPIN     BSR    SEND       PRINT CHARACTER
3006 8E   8000                    LDX    #$8000     SET DELAY
3009 8D   41                      BSR    DELAY
300B 48                           ASLA              MAKE NEW CHARACTER
300C 8D   0B                      BSR    SEND       PRINT NEW CHARACTER
300E 86   0D                      LDA    #$0D       SEND CARRIAGE RETURN
3010 8D   07                      BSR    SEND       PRINT IT
3012 AD   9F A000                 JSR    [POLCAT]   CHECK FOR KEY
3016 27   E8                      BEQ    START      DO AGAIN IF NO KEY
3018 39                           RTS               RETURN TO DEBUG MONITOR
                         * OUTPUT ROUTINE
3019 34   12             SEND     PSHS   A,X
301B 8D   2A                      BSR    SPACE      SEND START BIT
301D 9E   95                      LDX    BAUD       GET DELAY VALUE
301F 8D   2B                      BSR    DELAY      COUNT IT DOWN
3021 C6   08                      LDB    #8         BIT COUNTER
3023 34   04             NEXT     PSHS   B          SAVE COUNTER VALUE
3025 5F                           CLRB              SET ZEROS
3026 44                           LSRA              BIT 0 TO CARRY
3027 59                           ROLB              BIT INTO B
3028 59                           ROLB              ALIGN BIT WITH PORT
3029 F7   FF20                    STB    SEROUT     SEND CURRENT BIT
302C 9E   95                      LDX    BAUD       MAKE NEW DELAY
302E 8D   1C                      BSR    DELAY
3030 35   04                      PULS   B          GET COUNTER
3032 5A                           DECB              COUNT BIT SENT
3033 26   EE                      BNE    NEXT
3035 8D   0A                      BSR    MARK       SEND STOP BITS
3037 9E   95                      LDX    BAUD       FIRST STOP BIT
3039 8D   11                      BSR    DELAY
303B 9E   95                      LDX    BAUD       SECOND STOP BIT
303D 8D   0D                      BSR    DELAY
303F 35   92                      PULS   A,X,PC     RETURN FOR NEXT CHARACTER
3041 C6   02             MARK     LDB    #2         SET BIT 1 HIGH
3043 F7   FF20                    STB    SEROUT     TO GIVE STOP BIT
3046 39                           RTS
3047 5F                  SPACE    CLRB              ALL ZEROS
3048 F7   FF20                    STB    SEROUT     TO GIVE START BIT
304B 39                           RTS
304C 30   1F             DELAY    LEAX   -1,X       COUNT DOWN DELAY
304E 26   FC                      BNE    DELAY
3050 39                           RTS
                                  END    START

ERROR(S) DETECTED
```

Listing 2

Receive data from Serial Adapter.

```
                         * THIS PROGRAM WILL INPUT AN 8-BIT VALUE ON THE
                         * COLOR COMPUTER SERIAL PORT.
                         * EQUATES
                 00AB    DVAL     EQU    $AB        DELAY VALUE
                 0095    BAUD     EQU    $95        BUFFER FOR DELAY CONSTANT
                 FF21    CTLIN    EQU    $FF21      CONTROL PORT FOR SERIAL IN
                 FF22    PORTIN   EQU    $FF22      SERIAL IN PORT
                         * MAIN PROGRAM
3000                              ORG    $3000
3000 7F   3040           START    CLR    BUFR       CLEAN SLATE
3003 8E   00AB                    LDX    #DVAL      SET UP TIMER
3006 9F   95                      STX    BAUD
3008 108E 3041                    LDY    #BUFR+1    SET MEMORY POINTER
300C B6   FF22           READ     LDA    PORTIN     TEST RS232 BIT
300F 84   01                      ANDA   #1
3011 26   F9                      BNE    READ       SPIN UNTIL START BIT
3013 9E   95                      LDX    BAUD       GET DELAY VALUE
3015 1F   10                      TFR    X,D        DIVIDE BY TWO
3017 47                           ASRA
3018 56                           RORB
3019 1F   01                      TFR    D,X
301B 8D   1E                      BSR    DELAY      AND COUNT IT DOWN
301D B6   FF22                    LDA    PORTIN     STILL START BIT?
3020 84   01                      ANDA   #1
3022 26   E8                      BNE    READ       IF NOT, SKIP IT
3024 9E   95                      LDX    BAUD       OTHERWISE, GET FULL DELAY TO
3026 8D   13                      BSR    DELAY      READ MIDDLE OF FIRST BIT
3028 C6   08             SETUP    LDB    #8         BIT COUNT
302A B6   FF22           INPUT    LDA    PORTIN     READ PORT
302D A7   A0                      STA    ,Y+
302F 44                           LSRA
3030 76   3040                    ROR    BUFR       BIT INTO STORAGE
3033 9E   95                      LDX    BAUD       SET UP TIMER
3035 8D   04                      BSR    DELAY
3037 5A                           DECB              COUNT DOWN BITS
3038 26   F0                      BNE    INPUT      AND DO EIGHT PASSES
303A 39                           RTS               THEN QUIT
303B 30   1F             DELAY    LEAX   -1,X
303D 26   FC                      BNE    DELAY
303F 39                           RTS
3040                     BUFR     RMB    1
                                  END    START

0 ERROR(S) DETECTED
```

THE MOMENTUM™ GROUP

Presented by Gold Disk™

Featuring the finest in **educational and home utility** software for the Apple:

ANIMATION GRAPHICS: PAINT PROGRAM
Powerful yet easy-to-use shapes creator and animator for the novice or experienced graphics artist. Apple II or IIe, 48 K. 1 drive. $35.00

ANIMATION GRAPHICS: SUPER SLIDE SHOW
Powerful slide show creator for novice or expert. Apple II or IIe, 48 K. 1 drive. $29.00

INTRODUCTORY SPECIAL: NOW THROUGH JANUARY 31, 1984: Buy both the PAINT PROGRAM and the SUPER SLIDE SHOW together for one low package price of $45.00.

SCANDURA MICRO-TUTOR II SERIES (Grades 1-5)
Interactive and individualized instruction disks with hi-res graphics and the latest advances in cognitive theory. By Dr. Joseph Scandura, specialist in structural learning.

- **Complete Tutorial Arithmetic Series** (4 disks complete). A total system to pinpoint your child's needs in arithmetic and get results. Apple II or IIe, 48 K. I drive. $135.00. With Diagnostic Pre-Test Option, $160.00.
- **Complete Language Arts Series** (8 disks complete). Helps identify the language skills needed by your child and improves them dramatically. Apple II or IIe, 48 K. 1 drive. $400.00

LOCUS MICRO-TUTOR (All Grades)
Allows an educator to build a customized course for students on any subject matter with C.A.I. instruction disks. Can also print and store test materials. No programming knowledge necessary. Apple II or IIe, 48 K. 1 drive. $59.95.

LITMAS DATA SYSTEM
High-speed system for data storage, retrieval and statistical analysis. Written by Eric Sohr, M.D., LITMAS is the only cross-indexing data base for the Apple, with up to 512 keywords possible, and up to 2048 combinations of keyword phrases possible. LITMAS' speed is derived from its unique "bit map" structure. Delightfully-easy to use, excellent manual. Apple II or IIe, 48 K. 1 drive. $145.00

BRAHMAN HOME HEALTH DISK
Comprehensive medical information program. Written by an E.R. physician, this program can help you live healthier and possibly save someone's life. Factual, inclusive, consumer-oriented ... Covers emergencies, injuries, drugs, treatments, etc. 2 full disks with 15 programs. Apple II or IIe, 48 K. 1 drive. $79.95. Annual update available.

PRO—FORMA TAX PREPARER/TAX BREAK ANNUAL
On time and reliable since 1981 — never missed a year's update. A sophisticated tax optimizer which quickly determines the lowest tax by automatically taking into account every benefit, loophole, exclusion, adjustment and credit possible. User-oriented and very fast, typically taking 3 minutes from data input to return preparation. Program generates a detailed 7-page tax summary report indicating what form, line number and amount to be entered to obtain the very lowest tax. Also contains a unique "AUDIT POTENTIAL SCORE" which compares your return with that of a typical taxpayer with the same income, alerting you to any unusually high deductions that could trigger an audit. Apple II or IIe, 48 K. 1 drive. $129.95. Annual update available.

Distributed by MOMENTUM through your local dealer, or call national order agent for VISA/MC or C.O.D. order at **1-800-368-2260** in Maryland, 301-592-5949. Or write to MOMENTUM, 300 W. 31st St., Baltimore, Md. 21211. VISA MasterCard 30-DAY MONEY BACK GUARANTEE IF NOT SATISFIED!

Next Month in Micro

User-Defined Character Sets for the FX-80

Many of the line drawings and special characters appearing in this month's issue were printed on the new EPSON FX-80 printer using the "User-Defined" character capability. Since the FX-80 permits almost 256 different characters to be defined by the user, it can be used to print the character sets of the VIC-20, Commodore 64, PET, Atari and other micros which have built-in 'non-ASCII' characters. The user must, however, define the characters, and there is one great nuisance involved with creating them for the FX-80. The FX-80 uses basically an 8 by 6 array, but permits (with some restrictions), dots to be placed *between* the six columns. Using these extra five (5) *in-between* columns results in an 8 by 11 matrix which allows great characters, but makes the translation from the 8 by 8 more difficult, time-consuming, and frustrating.

Next month's MICRO will contain a chart of user-defined charac ers for the FX-80 corresponding to the standard graphic characters of the Commodore and Atari microcomputers. With this chart you will be able to readily use this powerful new printer to expand your systems' capabilities — without a lot of effort!

Electronic Spreadsheets

Electronic spreadsheets are helping many people in the business world, including the MICRO staff. How do these programs work? Can a spreadsheet help you in your business or at home? Next month we will take an in-depth look at available spreadsheets for your microcomputer — what functions they provide, the kinds of jobs for which they are useful, and a comprehensive listing of the spreadsheet packages currently available for our microcomputers. We will also have some spreadsheet reviews, sample spreadsheets from various programs, and some ideas on how to make better use of your spreadsheet software.

Micro Calc

The March 1983 issue of MICRO presented Loren Wright's Micro Calc program for Commodore computers and Phil Daley's version for Apple. Due to the popularity of the program, we decided to present a feature article on Micro Calc. Now it works on more computers, and we've added more features.

Micro Calc is an easy-to-use program that helps you make fast work of repetitive calculations. It can also help you understand BASIC's numeric functions. Once you have a screen constructed, it can be saved on tape or disk for future use.

The Commodore 64 version includes a separate comment field adjacent to each line. The VIC-20, Commodore 64, and PET versions all support multiple statements on a line and comments. All of these now support disk as well as tape. There is a new Atari version by Tom Marshall an a CoCo version by Phil Daley.

Finally, there will be many new screens to type in, save and use. Applications include finance, trigonometry, and metric conversion.

MICRO

Raise your Apple's IQ Twelve Times A Year!

A One Year Subscription Brings You 12 Issues With:

Over $500 of Programs for your Home, Business, Education and Entertainment. Complete Program Listings with Instructions.

Comprehensive Articles that show what each program does, how to use it and how to type it into your Apple, Franklin ACE or other Applesoft-compatible computer.

Regular Features for the Beginner and Expert.

On The Scene
The Latest New Software/Hardware Releases.

Products! Inside and Out
Comprehensive Product Reviews.

Education Corner
Programs that help make Learning Fun.

Tips 'N Techniques
Little known programming Tricks you can Use.

Disassembly Lines
An Expert reveals the mysteries of Applesoft.

Utilities
Superchargers for Basic, DOS, Printing, and More.

Games
Arcade Fun you can Type and Run.

Note
☐ Domestic U.S. First Class subscription rate is $51.95
☐ Canada Air Mail subscripton rate is $59.95
☐ Outside the U.S. and Canada Air Mail subscription rate is $89.95
All payments must be in U.S. funds drawn on a U.S. bank.

Try a NIBBLE!

Here's what some of our Readers say:

☐ *"Certainly the best magazine on the Apple!"*
☐ *"Impressed with the quality and content."*
☐ *"Programs remarkably easy to enter."*
☐ *"I'll be a subscriber for life!"*
☐ *"Your service is fantastic . . . as a matter of fact, I'm amazed!"*

Try a NIBBLE!

NIBBLE is focused completely on the Apple and Applesoft-compatible computers.

Buy NIBBLE through your local Apple Dealer or subscribe now with the Coupon or Order Card in this issue.

You'll want Back Issues Too!

Here are some examples of programs you can get:

The Investor—Stock Tracking, Reporting, and Graphing.

Recipe Box—Kitchen/Menu Management made Fun.

The Librarian—Auto Logging and Retrieval of your Disks.

Designer/Illustrator—Art/Design Creation and Composition with Graphics.

Machine Language Editor—Quick and Easy Aid for Typing and Changing M/L Programs.

And Much . . . Much More!

NIBBLE will become a permanent part of your Reference Library. Discover why 95% of NIBBLE Readers save every issue!

Join more than 120,000 Apple/Ace users who say:
"NIBBLE IS TERRIFIC!"

SUBSCRIBE NOW AND SAVE $12.00 OFF THE COVER PRICE!

nibble

MasterCard VISA
We accept Master Charge & Visa

Box 325, Lincoln, MA 01773 (617) 259-9710

I'll try nibble!
Enclosed is my $26.95 (for 12 issues)
(Outside U.S., see special note on this page.)

☐ **check** ☐ **money order** ☐ **bill me (U.S. only)**

Your subscription will begin with the next issue published after receipt of your check/money order.

Card # ______________ Expires ________
PLEASE PRINT CLEARLY
Signature ______________
Name ______________
Address ______________
City ______________
State ______________ Zip ________

Advertiser's Index

MICRO Magazine 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.

National Advertising Representatives

Home Office:
Sheila McDonough, Advertising Representative
10 Northern Boulevard
P.O. Box 6502
Amherst, NH 03031

West Coast:
The R.W. Walker Co., Inc.
Gordon Carnie
2716 Ocean Park Boulevard, Suite 1010,
Santa Monica, California 90405 (213) 450-9001

serving: Washington, Oregon, Idaho, Montana, Wyoming, Colorado, New Mexico, Arizona, Utah, Nevada, California, Alaska, and Hawaii (also British Columbia and Alberta, Canada)

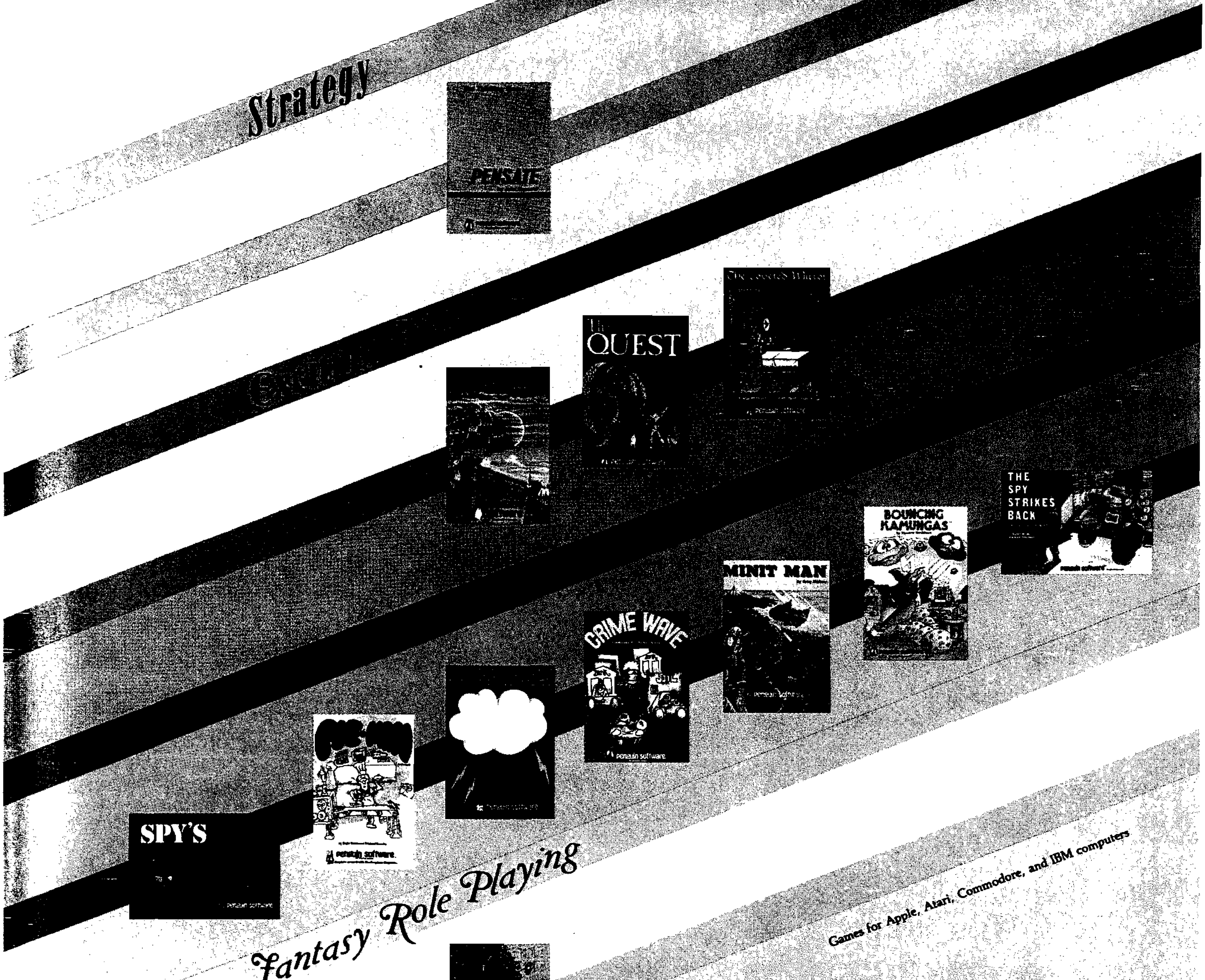
Mid-West Territory:
Thomas Knorr & Associates
Thomas H. Knorr, Jr.
333 N. Michigan Avenue, Suite 401
Chicago, Illinois 60601 (312) 726-2633

serving: Ohio, Oklahoma, Arkansas, Texas, North Dakota, South Dakota, Nebraska, Kansas, Missouri, Indiana, Illinois, Iowa, Michigan, Wisconsin, and Minnesota.

Penguin software™

the graphics pe

The Next Generation Is Here.



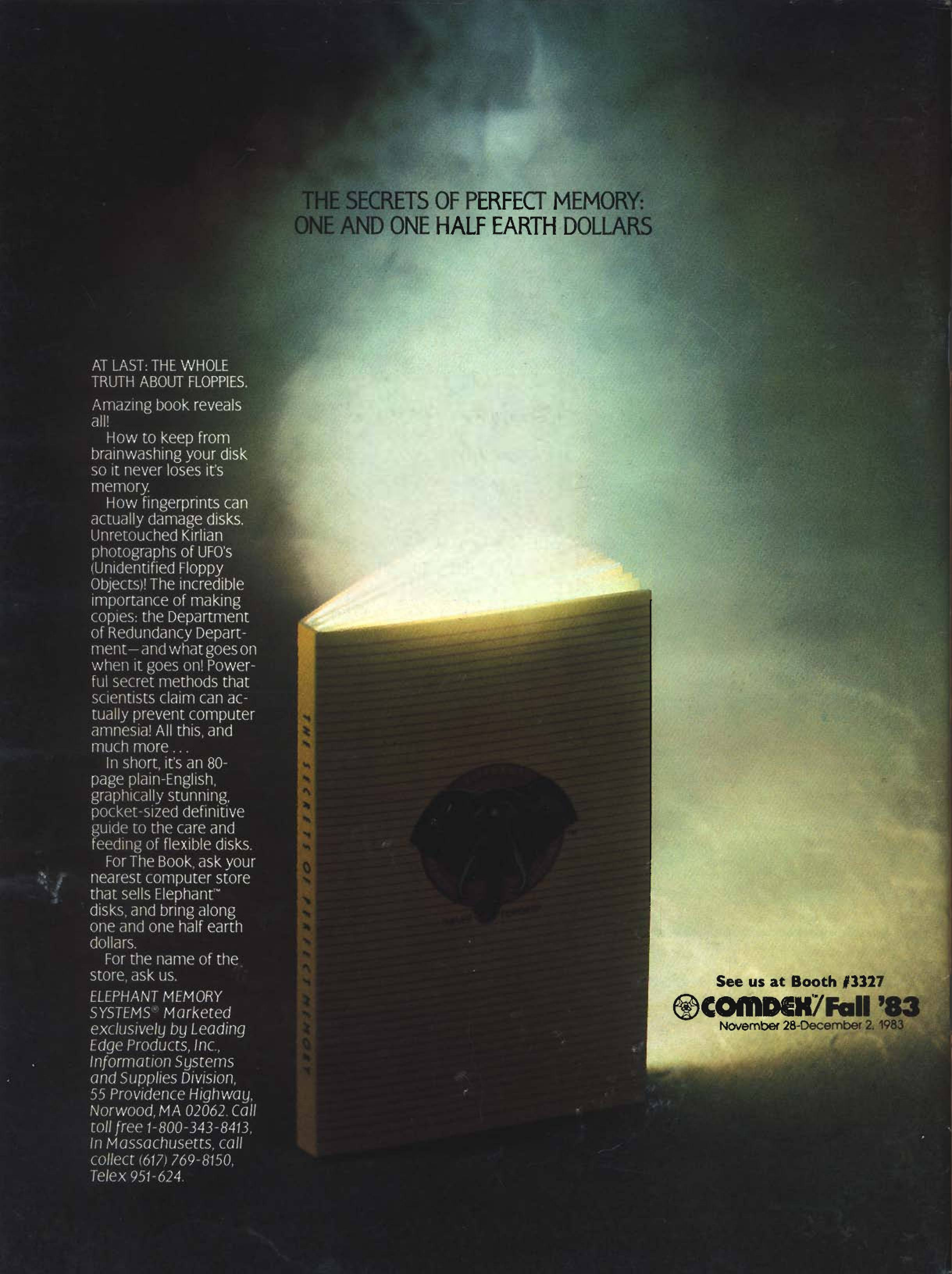
At Penguin, we don't rely on the same old game formulas, adding to the mish-mash of look-alike games already on the market. We look for creativity, originality, and innovation. Games like **Minit Man,** challenging you with two types of arcade action simultaneously; **Coveted Mirror,** an adventure with animation and arcade games throughout; **The Spy Strikes Back,** an arcade game with strategy and a touch of adventure; **Pensate** and **Tactic 9,** games that make you think; and **Expedition Amazon,** an intriguing and humorous fantasy game.

And Each Is Only $19.95! More Fun For Your Money From Penguin Software

Write for a free catalog

830 Fourth Avenue, Box 311 Dept. C Geneva, IL 60134 For information - Call (312) 232-1984 Dealer orders only - Call (800) 323-0116

Apple is a trademark of Apple Computer, Inc. Atari is a trademark of Atari, Inc. Commodore is a trademark of Commodore Business Machines, Inc. IBM is a trademark of International Business Machines Corp. Penguins are hard on pool toys.


THE SECRETS OF PERFECT MEMORY:
ONE AND ONE HALF EARTH DOLLARS
AT LAST: THE WHOLE TRUTH ABOUT FLOPPIES.
Amazing book reveals all!
How to keep from brainwashing your disk so it never loses it's memory.
How fingerprints can actually damage disks. Unretouched Kirlian photographs of UFO's (Unidentified Floppy Objects)! The incredible importance of making copies: the Department of Redundancy Department—and what goes on when it goes on! Powerful secret methods that scientists claim can actually prevent computer amnesia! All this, and much more . . .
In short, it's an 80-page plain-English, graphically stunning, pocket-sized definitive guide to the care and feeding of flexible disks.
For The Book, ask your nearest computer store that sells Elephant™ disks, and bring along one and one half earth dollars.
For the name of the store, ask us.
ELEPHANT MEMORY SYSTEMS® Marketed exclusively by Leading Edge Products, Inc., Information Systems and Supplies Division, 55 Providence Highway, Norwood, MA 02062. Call toll free 1-800-343-8413, In Massachusetts, call collect (617) 769-8150, Telex 951-624.
THE SECRETS OF PERFECT MEMORY
See us at Booth #3327
COMDEX/Fall '83
November 28-December 2, 1983