MICRO

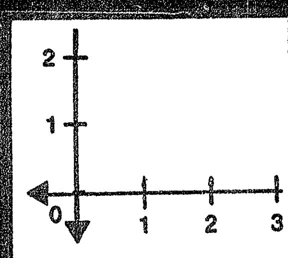
Plot Data With Character Graphics

Structured Game Design: Transform Imagination Into Graphic Displays

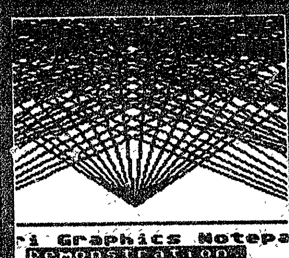
Directory Menu for Apple, Atari, C64, and Color Computer

Faster Math Operations in Microsoft BASIC

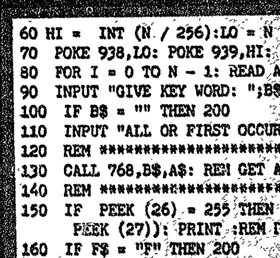
Atari Graphics Notepad in FORTH



See page 29



See page 69



```
60 HI = INT (N / 256):LO = N
70 POKE 938,LO: POKE 939,HI:
80 FOR I = 0 TO N - 1: READ A
90 INPUT "GIVE KEY WORD: ";B$
100 IF B$ = "" THEN 200
110 INPUT "ALL OR FIRST OCCUR
120 REM ********************
130 CALL 768,B$,A$: REM GET A
140 REM ********************
150 IF PEEK (26) = 255 THEN
    PEEK (27)): PRINT :REM P
160 IF F$ = "F" THEN 200
```

See page 40

Animated Graphics Routines for the 6809

Random Number Generator for the Apple

Review of Powerful Programmer's Utility for C64

MICRO™

Editor's Notes

This month's issue of MICRO introduces a new format, making it even more valuable and easier to read. Just turn to the Table of Contents for a peek. We've sectionalized the magazine by system — Apple, Commodore, Atari, and Color Computer. Now you have at least 10 to 15 pages of the magazine devoted only to your computer! Plus we still offer a general section with information for everyone.

We haven't altered MICRO's unique content or professional style. You can still count on MICRO for intermediate to advanced information on these systems — serious, useful information for serious computerists. And, since we've more carefully defined the systems we cover (eliminating the OSI, AIM, SYM, and KIM as discussed in August's Editorial), we are able to cover more completely the Apple, Commodore, Atari, and Color Computer systems.

We think you'll find, over the next few months, that MICRO is growing in quality. Our loyal, long-time readers will be pleased with the way MICRO continues to meet their expectations, and new readers will be pleasantly surprised at discovering a serious, useful, professional source of information.

October's Highlights

Our concentration in October is on programming techniques. We offer a directory menu for each system, plus many other helpful techniques and methods to improve your programming. In the general section Loren Wright provides a routine to plot data with character graphics for all the systems, and Michael Allen demonstrates structured game design.

In the Apple section look for "Rapid String/Substring Search," by L.S. Reich, a random number generator from Bill Walker, and a linear search technique by Richard Vile. The Commodore section brings you faster math operations in Microsoft BASIC (Peter Hiscocks), and a bank-switched JSR by Terry Peterson.

Atari users will find their directory menu, Paul Swanson's From Here to Atari column, and "Atari Graphics Notepad in FORTH," by Mike Dougherty. And, for Color Computer enthusiasts there is John Steiner's CoCo Bits column, a directory menu, and animated graphics routines from Craig Carmichael.

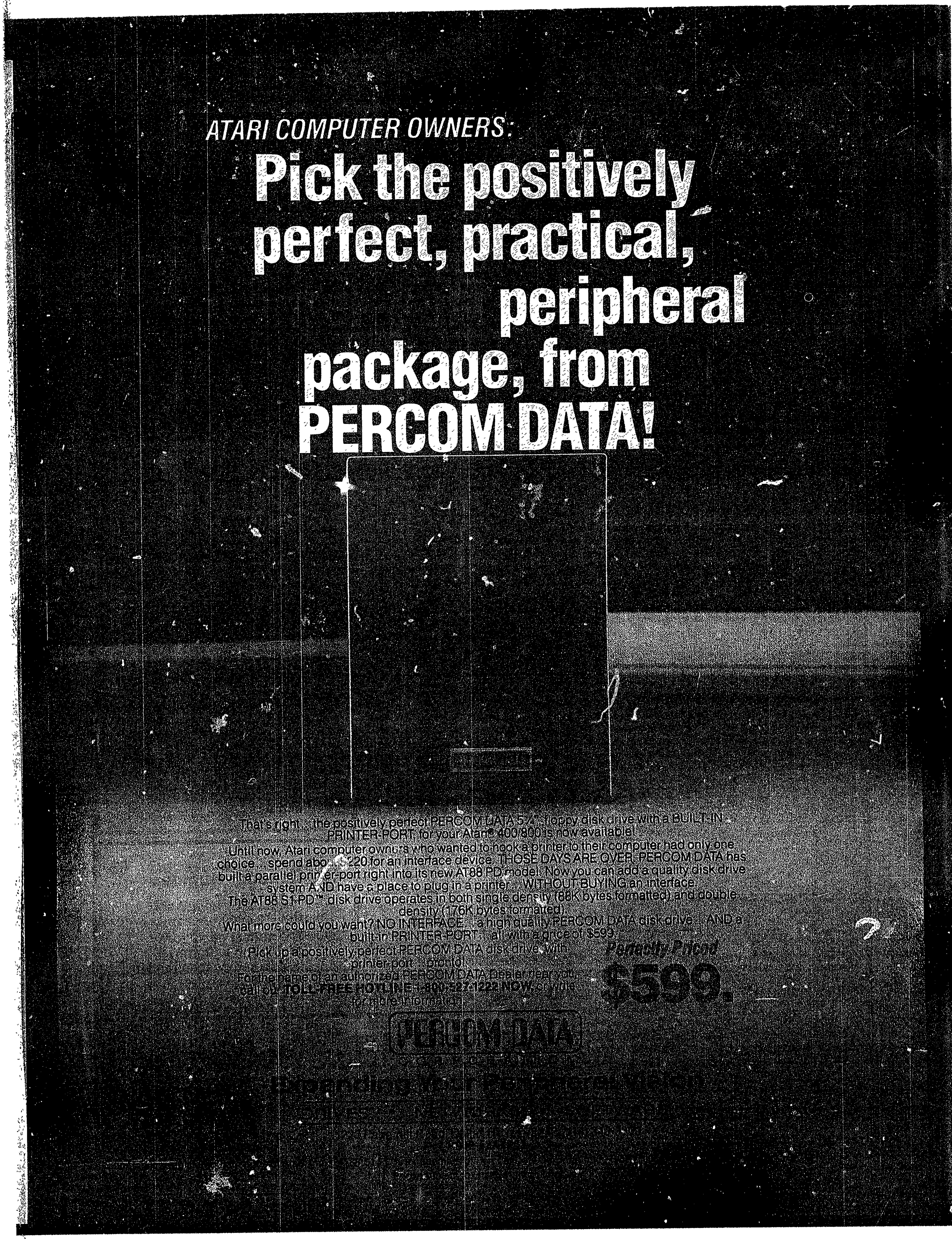
Each month the system sections will become more carefully organized; in November each will contain its own Software and Hardware Catalog and Reviews in Brief departments. We hope you enjoy our new format.

Enter Our Graphics Contest!!!

We're sponsoring an exciting contest for those of you interested in designing graphics pictures. You could win one of many prizes — big and small! Just use your favorite graphics program on your favorite microcomputer (either a Commodore, Apple, Atari, or Color Computer) and create! Turn to page 108 for all the details!

About the Cover.
The graphic this month was created on the C64 by Al Korostynski's directory menu program.

MICRO


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

MICRO™

Advancing Computer Knowledge

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

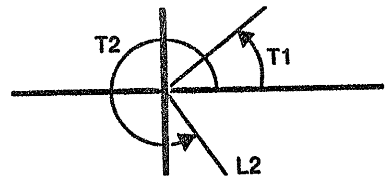
MICRO is published monthly by: MICRO, Amherst, NH 03031. Second Class postage paid at: Amherst, NH 03031 and additional mailing offices. USPS Publication Number: 483470. ISSN: 0271-9002. Send subscriptions, change of address, USPS Form 3579, requests for back issues and all other fulfillment questions 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.

Programming Techniques

25 Programming Techniques
Phil Daley
A brief discussion on useful techniques

26 The Directory Menu
Phil Daley
An introduction to this user-friendly utility

28 Plotting Data with Character Graphics
Loren Wright
General purpose routines for the C64, VIC-20, PET, Apple, Atari, and TRS-80C



34 Structured Game Design
Michael Allen
Transform imagination and fantasy into graphic or text displays

97 Interface Clinic
Ralph Tenny
Convert your serial I/O into parallel I/O using a UART

Apple

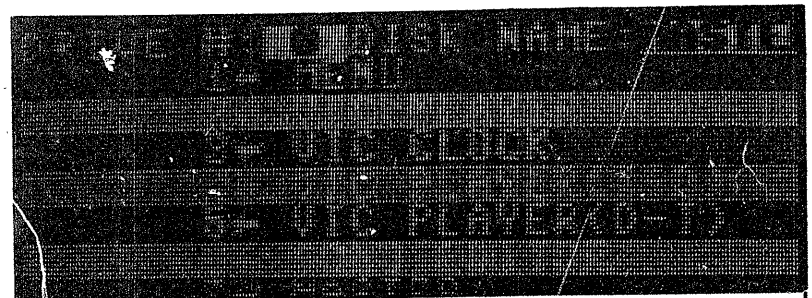
18 Apple Slices
Jules Gilder
Turn your IBM PC into an Apple

38 Rapid String/ Substring Search
L. S. Reich
Use a machine-language program to obtain a rapid array search.

42 Directory Menu for the Apple
Keith Davison and Phil Daley
Convert the normal CATALOG routine into a string array of individual entries

44 Random Number Generator
Bill Walker
Theoretical basis for generating uniformly distributed pseudo-random numbers

48 Linear Search
Richard Vile
Learn how to search through a list of numbers

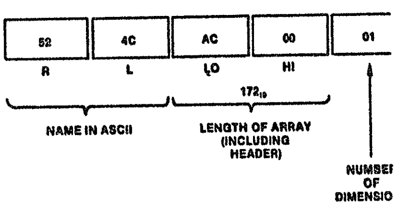


The Directory Menu...page 26

Commodore

12 Commodore Compass
Loren Wright
A look at POWER64 from Pro-Line Software

54 Faster Math Operations in Microsoft BASIC
Peter Hiscocks
Speed up array operations



58 Bank-Switched JSR
Terry Peterson
Use SuperPET's bank-switched memory with 6502 machine-language programs

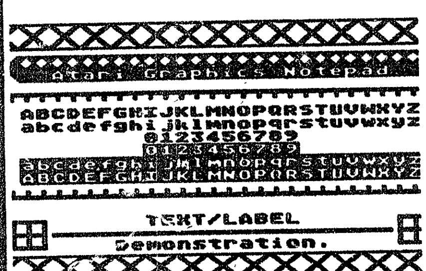
62 Directory Menu for the Commodore 64
Al Korostynski
A fast version of this utility

Atari

14 From Here to Atari
Paul Swanson
Information on the three GTIA modes and a new editor, EREDIT

34 Directory Menu for the Atari
Jerry White
All you need are your DOS master disk, a blank disk, and these two BASIC programs

66 Atari Graphics Notepad in FORTH
Mike Dougherty
Interactively construct images through simple keyboard commands



Color Computer

22 CoCo Bits
John Steiner
New disk ROMs, the J & M disk controller card, and Elite*Calc

76 Directory Menu for the Color Computer
Steve Kronschnabel and Phil Daley
This version offers a directory backup

77 Animated Graphics Routines for the 6809
Craig Carmichael
Display moving animated figures and custom characters

DEPARTMENTS

2 Editor's Notes
7 Letters/Updates
87 Hardware Catalog
90 Software Catalog
100 New Publications
102 Reviews in Brief
107 Next Month in MICRO
111 Advertiser's Index
112 News Page

ARK COMPUTING

P.O Box 2025
CORONA, CA 91720

CALL FOR NEW FREE CATALOG (714) 735-2250

BUS. SOFTWARE	LIST	ARK
Accounts Recievable	395.00	264.95
CPA General Ledger	250.00	165.00
CPA II Acct. Recievable	250.00	165.00
CPA IV Payroll	250.00	165.00
CPA V Property Mgmt.	495.00	329.95
Gen. Ledger/Payables	495.00	329.95
Payroll	395.00	264.95

COMPLIERS/ASSEM.	LIST	ARK
A.L.D.S.	125.00	89.95
Merlin	64.95	44.95
T.A.S.C.	175.00	129.95
Lisa v2.6 [illegible]ctor	79.95	59.95

INTERFACES	LIST	ARK
BPO 16K	179.00	119.95
CCS Async Serial	135.00	114.95
CPS Multifunction	239.00	169.95
Dumpling GX	159.95	99.95
Versacard	199.00	149.95

✱CP/M® is a registered trademark of Digital Research

MISC.	LIST	ARK
Gra Forth	75.00	49.95
The Home Accountant	74.95	49.95
Master Diagnostics +	99.95	69.95
S.A.M.	124.95	84.95
Sensible Speller	125.00	89.95

GRAPHIC	LIST	ARK
The Artist	79.95	59.95
G.A.S.	75.00	49.95
LPS II Light Pen	349.95	249.95

FORECASTING	LIST	ARK
Mathemagic	89.95	59.95
Multiplan	275.00	169.95
Supercalc II	295.00	194.95
Visicalc	250.00	179.95

LAZERWARE	LIST	ARK
Anix	49.95	39.95
Pascal	39.95	29.95
Lisa v2.6	79.95	59.95

80 COLUMN CARDS	LIST	ARK
Videx Combo	375.00	235.00
WIZ 80	249.95	164.95
Smarterm II	179.00	129.95

✱ Z80/CPM®	LIST	ARK
CPM Card	399.00	334.95
Premium Pak	685.00	494.95
Softcard	345.00	224.95
Softcard Plus	645.00	449.95
Z Card	169.00	134.95

BEAGLE BROS	LIST	ARK
Alpha Plot	39.50	29.95
Apple Mechanic	29.50	21.95
Beagle Bag	29.50	21.95
DOS BOSS	24.95	15.95
Double Take	34.95	24.95
Flex Text	29.50	21.95
Frame-Up	29.50	21.95
Pronto DOS	29.50	21.95
Tip Disk #1	20.00	14.95
Typefaces	20.00	14.95
Utility City	29.50	21.95

MEMORY EXP.	LIST	ARK
Microsoft 16K	99.95	79.95
RAM disk 128K	379.95	304.95
RAM disk 320K	999.95	804.95
Ramex 128K	497.00	359.95
RAM Plus 16K	189.00	139.95
ROM Plus	155.00	124.95
STB 128K	499.00	359.95
WIZ 16K	95.00	59.95

WORD PROCESSING	LIST	ARK
Magic Window II	149.95	99.95
Bank Street Writer	69.95	54.95
Pie Writer 80 Col.	149.95	109.95
Screenwriter II	129.95	79.95
Word Handler	199.95	134.95
Pro Easywriter Combo	300.00	199.95
Letter Perfect	149.95	99.95
Super Text	125.00	99.95

DATA BASES	LIST	ARK
Data Perfect	129.95	109.95
D Base II	695.00	499.95
DB Master	229.95	159.95
PFS	140.00	99.95
The General Manager	229.95	159.95
d Util	99.95	74.95
PFS Graph	140.00	99.95
PFS Report	125.00	89.95

MODEMS	LIST	ARK
Apple Cat II	389.95	289.95
Micromodem	379.95	289.95
Micromodem w/term.	409.95	309.95
212 Apple Cat	725.00	639.95

UTILITY	LIST	ARK
Watson	49.95	39.95
G.P.L.E.	64.95	44.95
E.P.F. IV	79.95	49.95
The Inspector	59.95	49.95
Locksmith	99.95	69.95
Quick Code	295.00	214.95
Routine Machine	64.95	44.95
Visicalc Preboot	49.95	39.95
Apple Writer Preboot	19.00	15.00

COMMUNICATIONS	LIST	ARK
Data Capture	89.95	59.95
Hayes Terminal Prog.	99.95	64.95
Pro Ascii	129.95	89.95

MISC. Enhancements	LIST	ARK
CCS Calendar/Clock	1[illegible]0.00	99.95
Enhancer II	149.00	119.95
Eprom Programmer	395.00	355.95
Koala Graphics Tablet	124.95	89.95
Kraft Joystick	64.95	49.95
P100 Sweet P Plotter	795.00	699.95
ROM Writer	175.00	139.95
Snapshot	119.00	94.95
Super Fan II	79.95	59.95
Super Fan w/Zenar Ray	109.95	89.95
Super Switcher	295.00	219.95
Videx Function Strip	7[illegible].95	59.95

WE ACCEPT VISA/MASTERCARD, PERSONAL CHECKS (ALLOW 10 DAYS TO CLEAR) OR COD ($2.00 CHARGE). PLEASE INCLUDE 3% FOR SHIPPING ($2.00 MIN.) OR 5% FOR BLUE LABEL ($3.00 MIN.). FOREIGN SHIPPING 10% ($5.00 MIN.). CALIFORNIA RESIDENTS ADD 6% SALES TAX. ALL ITEMS ARE NEW AND CARRY MANUFACTURERS WARRANTY. PRICES AND AVAILABILITY ARE SUBJECT TO CHANGE WITHOUT NOTICE.

MICRO™

Letterbox

OSI Readers Respond

Dear Editor:

It was with great sadness that I read your editorial in the August, 1983, issue proclaiming the death of OSI, AIM, SYM, and KIM. If you really mean not to accept any more articles on these micros, such an action is Oedipal, or at least patricidal, on your part and may well be expected to bring the historic retribution from the Fates. (Personally, I would have recommended the Furies.)

I believe you may have miscalculated the extent of your present (and future) readership concerned with these micros, perhaps due to some inadequacy of your sampling technique or to the reticence of many users to answer surveys, etc.

As an example of such a possibility, I would cite the fact that, although they have been rather slow to get going in this area, the American Association of Physics Teachers, which represents the vast majority of both college and high school physics teachers in the U.S., has presented workshops at each of its meetings over the past three years (and at numerous other times and locations), which have introduced more than three thousand teachers to the use of the KIM and SYM. In fact, it was only this year that their advanced interfacing workshop was entirely shifted from the KIM to the SYM. These activities are building a considerable base of competence in the laboratory application of single-board computers and are resulting in the appearance in the scientific literature of applications of these devices. Single-board micros are the essence of such interfacing applications, where it would be patently absurd to dedicate an entire Apple or Commodore 64.

I have been a subscriber to MICRO almost from its beginning and I have always appreciated its role in supplying information on the specialized microcomputers that were of greatest interest to my own applications in the educational field. Please don't abandon that unique position and become just another general computer magazine.

Charles D. Geilker, Chm.
Department of Physics
William Jewell College
Liberty, MO 64068

Dear Editor:

The dropping of just the OSI and ASK articles and focusing on just the most popular and future computers is a mistake. Obviously, you are trying to expand your audience — one of them is making more room in the magazine — you can reach an even larger audience by covering all types and brand of machines. That's one reason why *BYTE* and *Computers and Electronics* are so successful. Why limit yourself to just a certain group of machines? Don't get me wrong, I'm not condemning your decision. Seeing a magazine with a potpourri of very useful information become specifically directed to a limited audience is like watching the United States step down on the ladder of world importance.

I guess the theme of this letter is to remember that not everybody has a PET, Coco, Atari, TI 99/4A, or any other of the few machines that you do cover. One thing that seemed to be a message in the August editorial is that if one submits an article demonstrating a new use for the MC68000 processor specific to the ASK machines, you'll ignore it! Again, I ask: Why just cover certain machines? Doing this just contradicts the subtitle of the magazine: "Advancing Computer Knowledge". Maybe it should now read: "Restricting Computer Knowledge". How about editorials on the latest portables? This letter was written on a Kaypro 4. Others using portables will appreciate useful information on this machine. Or how about the new Heathkit, the IBM PC, or the Hewlett Packards? Put all this together and you have either a new section in the magazine or a special issue!

Well anyway, thanks for helping me get the most out of my machine — something that other publications just couldn't have done.

Timothy Hu
1601 E. Lincoln Way
Cheyenne, WY 82001

Atari FORTH Topics

Dear Editor:

I was pleased to see Mr. Dougherty's article on FORTH applications for the Atari (62:92). I would be even more pleased if he (or anyone!) could grapple with another FORTH-for-Atari issue: disk files. Since FORTH disk files are incompatible with other DOSes, including Atari DOS, an article on reading and writing Atari DOS files from FORTH would be a boon.

Also, I thought the magazine's name was "MICRO" because of the computers, not because of its typography. If an article deserves no better than unreadable microscopic print, should the article be printed at all?

Mr. Swanson's column has twice mentioned ways of controlling peripherals from the Atari joystick parts. The explanations have been terse and in expert's language, largely, I expect, because of the limits to space for a column. An article, however, would give him the chance to expand on the issue in terms understandable (or at least usable) by ordinary BASIC programmers. Example programs would be necessary, even if in tiny print!

Ronald Pitts
RD #5
Kittanning, PA 16201

Dear Editor:

Thank you for the article "An Inexpensive Joystick for the Apple II" (62:48). I made it, and it works (after figuring out the orientation of the connector pin diagram). One note: I found a 16-pin DIP Header at my local Radio Shack (Part no. 276-1980, $1.69 + tax).

Please, have more program listings for Apple; I originally subscribed to MICRO because it had good programs that I could enjoy. I'd also like to see more hints/techniques on programming — and, if you can find them, projects like the joystick.

Carl E. Serkland
507 Fontonett Ave.
Livermore, CA 94550

(Continued on next page)

Dear Editor:

I read in your CoCo Bits column that a new ROM version for the Color Computer was expected soon. ROM version 1.1 has been out for some time as I have had my CoCo for over six months and it contains 1.1. This can be determined by typing "EXEC 41175" then "ENTER". This will cause the ROM version to be printed to the screen.

The disk Scripsit and Spectaculator from Radio Shack are run by typing "RUN DOS" then "ENTER".

I know a lot of people at work with micros that used to laugh at the Color Computer and think of it as a toy, but no more. Some of them have even told me they wish they had purchased one instead of what they have.

Your magazine has a lot of good information in it. It would be great to see more on the Color Computer.

Brent Flemming
3rd Floor, Systems
700 Newport Center Dr.
Newport Beach, CA 92660

Newton-Raphson: Novel or Not?

Dear Editor:

I did not have the pleasure of reading "Extending Newton — Raphson's Method to Evaluate Complex Root" by P. P. Ong (56:71), but have just spotted Dr. Ong's letter in the July issue, wherein he mentions that the "extension (of the Newton-Raphson method) to include complex roots is novel."

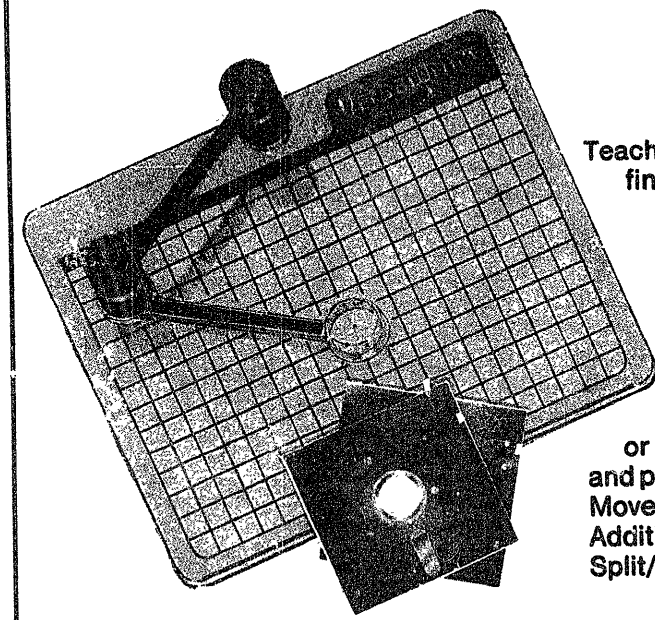
I am afraid it is not that novel at all. In my textbook, *Introduction to Numerical Methods*, published by the Macmillan Publishing Company in 1970, I use the Newton-Raphson method to solve an equation with complex roots. Though no prior instances of such use come immediately to mind, I doubt that the technique was new even back in 1968 when the book was written.

Peter A. Stark
P.O. Box 209
Mt. Kisco, NY 10549

(Continued on next page)

VersaWriter

FOR COMPLETE GRAHICS



GRAPHICS TABLET AND SOFTWARE

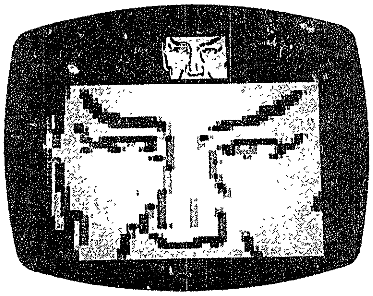
Teachers, artists, engineers, programmers & hobbiests find **VersaWriter** an easy to use tool for creating micro computer graphics. No programming experience is required. Pictures can be made by simply tracing. Even children can explore the exciting world of computer graphics. The **VersaWriter** is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

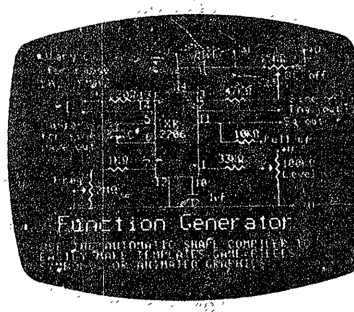
VersaWriter contains complete software for drawing with color, brushes & dots. Add text or fill in over 100 colors. Create your own shapes and place anywhere on the screen. Use Area/Distance, Move Picture, Electronic Drawing & Skeleton programs. Additional features include Reverse Picture, Scaling, Split/Full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

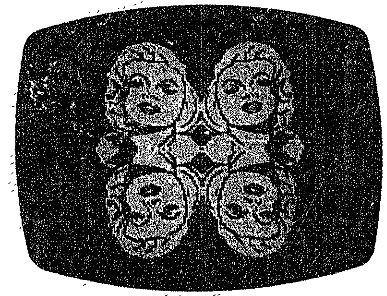
Complete Hardware/Software System for Apple II/II+/IIe **$299.00**

(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**

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**

*Requires VersaWriter

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

Letterbox *(continued)*

Dear Editors:

I refer to Peter Stark's letter to you concerning my article titled "Extending Newton-Raphson's Method to Evaluate Complex Roots" (56:71). I have already stated that NR's method is not new and numerous algorithms have already been tried long ago to extend it to complex roots (e.g., see W.E. Grove, *Brief Numerical Methods*, Prentice Hall, 1966, p. 9-14, as referred to in my article).

In Peter's book, the method described is, unfortunately, merely a simplified repetition of Grove's method and many others before him. Its chief defects, which render it unsuitable for a microcomputer (or even a more powerful computer), are:

(a) It necessitates a computer that can handle complex algebra, a requirement that is hard to satisfy even with present-day advanced microcomputers.

(b) Convergence is often so elusive and slow that it rarely concludes successfully.

(c) Perhaps most important of all, the algorithm suggested by Peter requires a lot of preliminary tedious (and error-prone) calculations, such as to rationalize all the term denominators. My program is universal and does not require any prior manipulation of complex numbers. Just enter the coefficients of the polynomial and the computer takes over completely.

Beset by all these defects, Peter's algorithm is probably only suitable as an academic topic. I hope his readers would not be enthusiastic enough to take up his advice (p. 116 second paragraph last line) that his method is worth trying in actual practice. As a lecturer myself I know that the surest way to stifle a student's interest in any subject is to disappoint him with unattainable expectations especially after considerable effort has been put in.

Was it just by pure chance that for a practical illustration of his method, Peter had chosen the most simple polynomial equation:

$$x^2 + x + 1 = 0$$

This example is almost too trivial for illustration on my program. It took barely five seconds to yield the answers:

$$x = -0.499999999 \pm 0.866025403i$$

on my Apple II+ computer, a result that is more accurate than Peter's. Such an oversimplified case certainly cannot be regarded as typical. Had Peter tried with even slightly more complicated examples, he would have realized the limitations of his method. I am quite sure that Peter's method is unlikely to be successful with either of the examples I used in my article.

I will leave it to your readers to judge whether my method is superior. The best way for Peter to support his contention that my algorithm is not novel is to cite some earlier paper describing the de Moivre extension of NR's method that I have proposed.

Dr. P.P. Ong
Physics Department
National University of Singapore
Kent Ridge
Singapore, 0511.

Send your comments, criticisms, or ideas to Letterbox, MICRO magazine, P.O. Box 6502, Amherst, NH 03031.

MICRO

MICRO™

Commodore Compass

by Loren Wright

POWER64 Makes Programming Easier

How would you like to be able to hit a single key on your Commodore 64 and have "POKE 53281," appear instantly on the screen? This "instant phrase" feature is only one of the many capabilities of POWER64.

To start, as soon as the POWER64 system is loaded in, many of the keys have new functions. For instance, the shifted "R" key immediately types "RETURN" on the screen, and the Commodore key and the "L" key cause "LEFT$(" to be printed. Also, the program listing behaves differently. Just move the cursor past the top or bottom of the screen and the listing begins to scroll, line by line. There is no longer any need to keep your finger poised nervously over the STOP key for fear of missing that vital line.

There is an AUTO command that automatically prints the next line number as you hit RETURN. If you quit the sequence (by hitting RETURN on the line with only a line number), you can resume programming where you left off simply by typing AUTO again. Of course you can specify the increment.

The DELETE command allows you to delete a whole range of lines with a single command — no more one-by-one line deletions! The RENUM command is the most powerful I've seen. It actually lets you renumber parts of your program, which means you can keep all your subroutines beginning on even thousands!

So much for the everyday commands. There is a lot more. As I mentioned at the beginning, you can redefine any key to a phrase, which could even be something like "FOR I=1 TO 100: ?I: GOSUB 2000: NEXT I". When you hit the redefined key (usually a shifted one), the whole phrase is instantly typed on the screen for you. You can even redefine a key to execute a whole subroutine consisting of many lines.

POWER64 has one of the most powerful search and search-and-replace packages I've seen: there are wild characters, a whole-program or next-occurrence operation, and a convenient repeat. Even the syntax is easy to remember.

Next, there's a whole set of debugging commands. The star of the show is the TRACE command, whose options include displaying at the top of the screen or in-line and full-line displays with variable value or just line numbers. After you invoke the TRACE command and type RUN, you can single step, trace continuously, or even execute for a while without any trace display. Responses to GET and INPUT statements are handled easily. The WHY command lets you know where in the program line an error occurred, and the DUMP command displays the values of all variables (except arrays). The DUMP display is such that you can easily change the value of any variable and CONTinue program execution.

The FIX and PTR commands restore the BASIC pointers that get fouled up, especially when you have loaded a machine-language program to a strange place. FIX also reinitializes POWER64. EXEC is a very powerful command that allows you to turn control over to a logical file.

TEST lets you LOAD (or type in) and RUN a second program without destroying the first. BACK sends you back to the first program.

That just about covers the capabilities of POWER64, but there is more to be considered. First, POWER64 is relocatable and automatically adjusts itself to the current Commodore 64 memory configuration. That means that, if you have a cartridge in place that takes memory from the top of BASIC, POWER64 will load in automatically below. (It does work with the C64-Link cartridge.)

Brad Templeton, the author of POWER64 and POWER (for the PET, reviewed in MICRO 50:69), has written a convenient resident assembler called PAL, which works extremely well with POWER64. The PAL assembler will be reviewed in a future issue of MICRO.

The manual, by Jim Butterfield and Brad Templeton, does well at both teaching the newcomer how to use the product and serving as a good reference. There are several appendices that help the advanced programmer get even more from POWER64.

It is easy to add your own commands and default-key definitions to POWER. In fact, a disk-oriented expansion package called MOREPOWER is included on the disk. MOREPOWER makes most disk tasks considerably easier. A single key will get you the disk directory or the disk-error messages. You can LIST a program (or even a sequential file!) from the disk without destroying your current program. There are SIZE and START commands that can read the length or beginning (for ML programs) of any program file on disk. You can LOAD or LOAD and RUN a program from a directory listing by simply moving the cursor to the left of the entry and hitting a single key. You can MERGE one BASIC file into another. (This is also possible with the EXEC command, but more is involved.) There are several others.

To prove that I haven't been paid off for this glowing review, I'll mention a couple of negative items. The AUTO command works in an annoying manner (at least for me). When you go to a previous line to make a change, the system doesn't recognize that you are no longer cranking out lines in the expected sequence. Other AUTO commands I've seen can handle this.

The price ($99.95) is certainly fair for all the things you get, but you should consider whether you really need everything POWER64 has to offer. Unless you do a fair amount of programming, the answer may very well be no. The manual mentions a cartridge version, which would make using the package more convenient, but less flexible. This is the only practical way that people without disk drives could use it.

POWER64 is available from Pro-Line Software Ltd., 755 The Queensway East, Unit 8, Mississauga, Ontario L4Y 4C5 (416-273-6350). MICRO

MICRO™

From Here To Atari

Paul S. Swanson

Several inquiries came in concerning the listing for the Mode 10 Painter Program, Part I, appearing in the July issue of Micro. The listing was typeset, which is a difficult format to decipher into the proper keystrokes when translating the graphics characters onto the Atari computer. The screen format listing in September should help you find any bugs. To make your screen conform to the 40-column format of these listings, POKE 82,0 before entering the program. That will set your left margin at column zero and give you a full 40-column screen.

GTIA Modes

The painter program also inspired a few questions about the three GTIA modes. All three of the modes are variations of GRAPHICS 8 screens. The shadow of the hardware register PRIOR, which is located at (decimal) 623, contains two switches that control the four possible interpretations. Since GRAPHICS 8 and GRAPHICS 0 screens both use the same type of interpretation, this location can also be used to control a GRAPHICS 0 screen so that it can be used as a GTIA character graphics screen.

Specifically, the interpretation for the GTIA modes all use four bits, so each byte contains the information for two pixels. If GRAPHICS 9 is declared, or a POKE 623,64 is executed, each four bits will be interpreted as the luminance of the hue contained in color register four (SETCOLOR 4,H,0 where H is the background hue). The luminance should be set to zero. If it is not zero it will create some undesirable effects on the display and at least some of the selections of luminances will be lost.

GRAPHICS 11 is the inverse of GRAPHICS 9. This interpretation happens in response to either a GRAPHICS 11 statement or POKE 623,192. All 16 hues will appear at the luminance stored with SETCOLOR 4,0,L (L is the luminance). The luminance of the black background is always zero, independent of the luminance selected. Using a hue number other than zero alters the background color, but has an undesirable effect on the hues as setting the luminance has on the luminances of a GRAPHICS 9 screen.

GRAPHICS 10, which corresponds to a POKE 623,128, is the only one that uses color registers other than register four. All nine color registers, which include the ones altered with SETCOLOR 0,... through SETCOLOR 4,... and the four used for the players and missiles (POKE 704,.. through POKE 707,..) are used on this screen. The allowed color values on the screen are zero through nine. The other seven are not practical to use, although they will produce colors. The screen background is taken from player zero's color and can be set with a POKE 704,HUE*16+LUMINANCE. POKE 705 through 707 for the colors one through three. SETCOLOR 0,... through SETCOLOR 4,... control colors four through nine. Notice that although SETCOLOR 4,... sets the background color in most other modes, it is a foreground color in mode 10. Mode 10 is the only mode that uses location 704 for the background color. The colors set with SETCOLOR 0,... through SETCOLOR 4,... are, in the same order, stored at locations 708 through 712, so POKE 704+COLORREG,HUE*16+ LUMINANCE, where COLORREG is the color number to set, HUE is the hue (0 through 15) and LUMINANCE is the luminance value (even numbers in the range 0 to 15).

EREDIT

EREDIT, by EHR3, Inc., is a newly introduced editor that can be used to create and/or edit BASIC or assembly-language files. It will also save all of the changes made to the file and allow you to "back out" of changes you have made, all the way back to the original version if you wish. It uses the files on disk as BASIC LIST files, or the equivalent in assembly language (a TYPEd file from the Synassembler, for example).

All of the screen editing controls normally used when entering or editing a BASIC program are available in this editor (i.e., the four arrow keys, INSERT, DELETE, etc.). It is also possible to be editing one file and list another file on the screen. Combining this with the full screen editing allows you to take lines, with or without additional editing, directly from another file into the file you are editing.

EREDIT is controlled by 31 commands. Several of them are DOS commands, allowing operations like file delete, file copy, directory list, disk format, lock and unlock, to be performed without entering DOS. Several commands allow statements to be relocated in the program text and renumbering is available. There is also an overlay option that prevents you from redefining a line. This can be very valuable in preventing accidental deletions of lines in the program being edited. The COMPARE command allows line-by-line comparison between programs, noting every line that is different in comparing two files (including lines appearing in one file but not the other).

There is no BASIC syntax checking in EREDIT so it is possible to create and/or edit BASIC files only to find errors while ENTERing it into BASIC later. This is not too

(Continued on next page)

From Here to Atari *(continued)*

serious a drawback, since the corrections are easily done from BASIC at that point. The only other drawback I noted was the fact that there was no immediate mode capability, or anything equivalent. Generally, when I am writing or editing a program, I use the immediate mode to execute ASC (– CHR$(conversions or perform calculations.

The user manual is in the reference book style, listing all of the commands in alphabetical order with adequate descriptions of each. Also, if the EREDIT disk is in the disk drive, the command HELP followed by a command name will list a brief description of the identified command on the screen. In both the manual and on the screen, the defaults are listed with the descriptions.

In summary, EREDIT can be a valuable tool to use in the development of software in BASIC or assembly language. It is available at EHR3, Inc., 174 Summit Avenue, Summit, NJ 07901. Suggested retail price is $49.95. Inquiries are invited at that address by mail or by telephone at (201) 277-6785. It is compatible with the Atari 1200XL computer as well as the Atari 400 and 800 computers.

Next Month

I will review two software packages in next month's column. One is XBASIC from SUPERware, a utility program that stays in memory when a BASIC program is being written and executed. It provides many interesting capabilities not available in the standard Atari BASIC relating to arrays, strings, player/missiles, and DOS. The other program is S.A.M. (Software Automatic Mouth) from Don't Ask Software, which adds a sophisticated speech synthesizer to your Atari with no additional hardware. These topics will be presented along with a simple way to implement a display-list interrupt, which can be used to change character sets, colors, or the switches that control the GTIA interpretation so that a text window may be added to a GTIA screen.

You may contact Paul at 97 Jackson St., Cambridge, MA 02140.

MICRO

MICRO™ Apple Slices

by Jules Gilder

Apple's Lisa computer is in the news again this month along with the Russians, typesetting by computer, and an alternative to hard disk drives. Those of you who have been following developments in computer hardware during the last six months are no doubt aware of the sheer joy expressed by IBM PC owners when they found out that they could turn their $4000 IBM PC into a $1500 Apple by simply plugging a card into the IBM computer. The much larger software base of the Apple II computer makes this $680 accessory quite attractive, and it is capable of running about 98% of all Apple software. (*Editor's note:* MICRO has a review copy of the Quadram *Quadlink*. It appears to run any software that does not require any specific hardware other than a printer. It emulates an Apple II as well as the Apple III does.)

Now it looks like the shoe is going on the other foot. According to industry sources, Apple is considering the possibility of making MS-DOS, the operating system used on the IBM PC, available on the Lisa computer. Since Lisa uses a 68000 microprocessor, it will be necessary to add an 8088 or 8086 microprocessor to the Lisa in order to let it run MS-DOS. The Unix operating system is also being developed for the Lisa, and speculation is that Concurrent CP/M-86 is being considered too.

The rush to get more operating systems working on the Lisa seems to be Apple's reaction to IBM's dominance of the office environment. Because of the small initial market for the $10,000 Lisa, the development of software for it has lagged seriously and only a small number of outside programs has been developed. Allowing the Lisa to run MS-DOS would open up the possibility of permitting the Lisa to read and write IBM disks directly. The only possible problem here is that Apple is using a specially designed disk drive with the Lisa and it might not be able to read the IBM disks.

Russian Apples

The Apple II computer is so popular that Apple has been having a lot of trouble with counterfeit computers being manufactured in the Far East. Now it seems that the Russians are getting into the act, too, and are producing their own counterfeit Apples. Unlike the counterfeits from Taiwan, the Russian clone is not expected to surface in the United States but will instead be going to schools and research centers inside Russia.

Setting Type with Your Apple

Many companies have discovered that they can use a word-processing program on their Apple computer to prepare material for a typesetting machine. The text is then sent by modem to the typesetter and camera-ready copy can be produced. Preparing this text with a word processor, while acceptable, is not really a good way to do things because word processors do not have all of the capabilities of a typesetting system, and it's almost impossible to tell what the final copy will look like. But now a small California company called The Type Source has developed a program that turns an Apple //e with a CP/M card into a full-blown typesetting terminal.

The program, known as STL (for Simplified Typesetting Language), sells for $695 and can be used to interface directly with a wide variety of phototypesetting machines including those from Compugraphic, Mergenthaler, APS, AM Varityper, and Itek. By permitting the user to enter all of the standard typesetting commands and giving him a display that is similar to those found on regular typesetting terminals, it is no longer necessary for the typesetter to get involved and translate the user's commands into typesetting commands. The net result is a saving of 30% to 50% in typesetting costs. The program is capable of working with up to eight different type faces at once and can produce very small 5.5-point characters or extra large 74-point characters. The package comes with a special telecommunication package for typesetters, a special HELP menu, and complete documentation including basic lessons in typesetting terminology and functions. STL will be available for the Apple //e in November

An Alternative to Hard Disk Drives

For those of you who have been thinking of buying a hard disk drive, take a look at the V1200 disk drive from Vista before you do. While this is not a hard drive, it is a system that can easily store up to 6 megabytes of data. The system uses a special cartridge that holds five removable 5¼-inch floppy disks. Each disk stores up to 1.2 megabytes of data on it. While each of the floppies looks a lot like the standard Apple floppy, looks can be deceiving. The storage capacity of these disks is equal to that of double-sided, double-density 8-inch floppies.

The drive used in the V1200 is a special one that is manufactured only by a company called Amlyn. In use, the V1200 selects the disk you specify, pulls it away from the others in the pack with a picking arm, and feeds it into the drive mechanism. While this may seem like a Rube Goldberg-type of device, in practice it works very well and in six months has never failed to operate properly. Because the angle of the cartridge in the drive changes with the disk selected, and the disk has to be loaded into the drive, initial access to data is a little slow. But once the disk is loaded, access to any of the 1.2 megabytes of data on it is as fast as that of a hard disk drive. That's because it loads data into the Apple the same way hard drives do — by using direct memory access (DMA), a technique that bypasses

(Continued on page 20)

wabash® diskettes for as low as $1.29 each!

Now get Wabash Quality at a CE Price

For over 17 years, Wabash has been making high quality and dependable computer products. Wabash diskettes are made to provide error-free performance on your computer system because every diskette has been totally and hypercritically tested. Since you can now buy Wabash computer products directly from CE, the world's largest distributor of magnetic media, you can now get maximum savings on every order. You can even order toll-free.

New Wabash Six Year Warranty

The quality of Wabash diskettes is stressed throughout the entire manufacturing process. After coating, all Wabash diskettes go through a unique burnishing process that gives each diskette a mirror-smooth appearance. Wabash then carefully applies a lubricant that is specially formulated to increase diskette life. Then, to keep out foreign particles, a unique heat seal bonds the jacket and liner together to help prevent contamination. After 100% hypercritical testing and certification, Wabash then packages each diskette, (except bulk pack) in a super strong and tear resistant Tyvek® evelope. The final Wabash product is then shrink-wrapped to insure cleanliness and reduce contamination during shipment. Wabash diskettes are so very reliable that Wabash now offers a six year warranty in case of defects in materials or workmanship on all diskettes purchased directly from Communications Electronics.

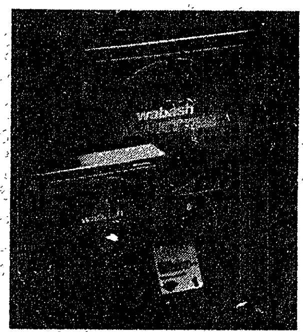
New...Wabash Diskette Duplication Services

Communications Electronics has teamed up with Wabash to provide a single-source solution for the diskette duplication requirements of software developers, OEM's and distributors. All service is in-house, to give you fast, dependable service. In most cases, delivery can be completed in five days. Whether you require 100, 1,000, or 10,000 copies per week, call CE first for a no obligation price quote. For additional information, please write us on your letterhead with your requirements.

SAVE ON WABASH DISKETTES Product Description	Part #	CE quant. 100 price per disk ($)
8" SSSD IBM Compatible (128 B/S, 26 Sectors)	F111	1.89
8" SSSD Shugart Compatible, 32 Hard Sector	F31A	1.89
8" SSDD IBM Compatible (128 B/S, 26 Sectors)	F131	2.39
8" DSDD Soft Sector (Unformatted)	F14A	2.99
8" DSDD Soft Sector (256 B/S, 26 Sectors)	F144	2.99
8" DSDD Soft Sector (512 B/S, 15 Sectors)	F145	2.99
8" DSDD Soft Sector (1024 B/S, 8 Sectors)	F147	2.99
5¼" SSSD Soft Sector w/Hub Ring	M11A	1.49
5¼" Same as above, but bulk pack w/o envelope	M11AB	1.29
5¼" SSSD 10 Hard Sector w/Hub Ring	M41A	1.49
5¼" SSSD 16 Hard Sector w/Hub Ring	M51A	1.49
5¼" SSDD Soft Sector w/Hub Ring	M13A	1.79
5¼" Same as above, but bulk pack w/o envelope	M13AB	1.59
5¼" SSDD 10 Hard Sector w/Hub Ring	M43A	1.79
5¼" SSDD 16 Hard Sector w/Hub Ring	M53A	1.79
5¼" DSDD Soft Sector w/Hub Ring	M14A	2.69
5¼" Same as above, but bulk pack w/o envelope	M14AB	2.49
5¼" DSDD 10 Hard Sector w/Hub Ring	M44A	2.69
5¼" DSDD 16 Hard Sector w/Hub Ring	M54A	2.69
5¼" SSQD Soft Sector w/Hub Ring (96 TPI)	M15A	2.59
5¼" DSQD Soft Sector w/Hub Ring (96 TPI)	M16A	3.69
5¼" Tyvek Diskette Envelopes - Price per 100 Pack	TE5	12.00

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

Quantity Discounts Available



Wabash diskettes are packed 10 disks to a carton and 10 cartons to a case. The economy bulk pack is packaged 100 disks to a case without envelopes or labels. Please order only in increments of 100 units for quantity 100 pricing. With the exception of bulk pack, we are also willing to accommodate your smaller orders. Quantities less than 100 units are available in increments of 10 units at a 20% surcharge above our 100 unit price. **Quantity discounts** are also available. Order 500 or more disks at the same time and deduct 1%; 1,000 or more saves you 2%; 2,000 or more saves 3%; 5,000 or more saves 4%; 10,000 or more saves 5%; 25,000 or more saves 6%; 50,000 or more saves 7%, 100,000 or more saves 8%, 500,000 or more saves 9% and 1,000,000 or more disks earns you a 10% discount off our super low quantity 100 price. Almost all Wabash diskettes are immediately available from CE. Our efficient warehouse facilities are equipped to help us get you the quality product you need, when you need it. If you need further assistance to find the flexible disk that's right for you, call the Wabash compatibility hotline. Dial toll-free 800-323-9868 and ask for your compatibility representative. In Illinois or outside the U.S. dial 312-593-6363 between 9 AM to 4 PM Central time.

Buy Wabash Diskettes with Confidence

To get the fastest delivery from CE of your Wabash computer products, we recommend you phone your order directly to our Computer Products Division and charge it to your credit card. Be sure to calculate your price using the CE prices in this ad. Written purchase orders are accepted from approved government agencies and most well rated firms at a 30% surcharge for net 30 billing. For maximum savings, your order should be prepaid. All sales are subject to availability, acceptance and verification. All sales are final. All prices are in U.S. dollars. Prices, terms and specifications are subject to change without notice. Out of stock items will be be placed on backorder automatically unless CE is instructed differently. Minimum *prepaid* order is $50.00. Minimum *purchase order* $200.00. All shipments are F.O.B. Ann Arbor, Michigan U.S.A. No COD's please. Non-certified and foreign checks require bank clearance.

For **shipping charges** add $8.00 per case or partial case of 100 8-inch flexible disks or $6.00 per case or partial case of 100 5¼-inch mini-diskettes for U.P.S. ground shipping and handling in the continental U.S.A.

Mail orders to: Communications Electronics, Box 1002, Ann Arbor, Michigan 48106 U.S.A. If you have a Visa or Master Card, you may call and place a credit card order. Order toll-free in the U.S. Dial 800-521-4414. In Canada, order toll-free by calling 800-265-4828. If you are outside the U.S. or in Michigan dial 313-994-4444. Telex anytime 810-223-2422. Order your Wabash diskettes today.

 Ad #U12483

MEMBER DIRECT MAIL/MARKETING ASSOCIATION

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

wabash error-free diskettes

COMMUNICATIONS ELECTRONICS™

Computer Products Division

813 Phoenix □ Box 1002 □ Ann Arbor, Michigan 48106 U.S.A.
Order TOLL-FREE 800-521-4414 or outside U.S.A. 313-994-4444

the microprocessor and loads data directly into memory.

The V1200 has a lot of benefits over the conventional hard disk drive. You don't have to worry about head crashes, drive alignments, lost data or backups. Because each disk in the cartridge can be easily removed and replaced, you can make your backups at the same time you save the original. One of the biggest advantages of the V1200, however, is the fact that it uses removable media. When a 5-megabyte hard disk fills up, you either have to purge old files or go out and buy a new 5-megabyte hard disk drive. With the Vista V1200, all you have to do is remove your 6-megabyte cartridge and insert a new one. Cartridges are fairly cheap and range in price from $60 to $70 for 6 megabytes of storage. The ability to have multiple cartridges also means that you can dedicate individual cartridges to particular applications if you want.

The V1200 comes with a disk that contains several utility programs on it. One of them is a program that modifies normal 3.3 DOS so that it can work with the Vista drive. The modification makes it possible for DOS to recognize five disk drives per slot. Thus, to access any of the five diskettes in the cartridge you simply add a D1 to D5 designation after the file name. The INIT command in DOS is disabled by the modification because a special formatting program is required to format V1200 disks. Another program supplied on the disk is one that modifies FID so that it works with the new drive. You have no idea how great it is to run FID, select the FREE SPACE option, and find out that you have 4500 sectors free (on a fresh disk). Another program provided with the drive is *Quickcharge.* This is a fastDOS program that significantly speeds up the reading and writing of DOS files.

In addition to working with standard DOS, the V1200 also works with the Pascal and CP/M operating systems. While the drive can be purchased from Vista for $1549, you can get it a lot cheaper ($1295) from A.P.P.L.E in Kent, WA.

You may contact Mr. Gilder at REDLIG Systems, Inc., 2068 79th St., Brooklyn, NY 11214.

MICRO

CoCo Bits

John Steiner

There have been big developments in the CoCo world — you may have already seen the newest color computers. They sport new ROMs and the top-of-the-line version 64K color computer has a deluxe keyboard. This option will be available as an add on for the older CoCos.

If you live around the Dallas-Ft. Worth area, you can attend another Rainbowfest. This CoCo convention will take place the weekend of October 14, 15, and 16. Next month, November 4, 5, and 6, Los Angeles, CA will be the site of Color Expo sponsored by Color Computer Magazine. In February, another Rainbowfest will be held in Newport Beach, California. I am going to try to make it to the Dallas Rainbowfest, if at all possible, and will be looking forward to meeting you at the show. If you see me, stop and say hi.

In the June issue, I included a listing for a program that will find the start, end, and execute addresses of a machine-language program, but neglected to point out that the routine works only on a tape system. If you use the routine with a disk system connected, the results are invalid.

A Look At The New Disk ROMs

The 1.1 disk ROMs are out, and the worst has proven to be true. Of all the disk software I have, the only software with disk I/O working under the new ROMs is Tandy-written. The only exception is disk Colorcom/E by Eigen Software Systems. I/O seems to be OK on that excellent terminal program, but I haven't had time to test it thoroughly. An acquaintance with Nelson's Colorterm software says that the software works well with the new ROM. I am writing this column using Telewriter-64, which has been my standard word processor, but I am having to use the 1.0 disk card as Telewriter is not compatible with the new ROM.

I am waiting for documentation on the new ROM capacity, but so far none is available. It is capable of loading a DOS from the disk, a definite improvement over the older ROM. Hopefully Tandy will document this ROM a little better and allow other software developers to have greater access to standard I/O routines. There will be a shake up in the support industry as companies scramble to convert their disk I/O to run on the new ROMs.

I chatted with John Waclo of Elite Software, who has recently released Elite*Calc. They will be converting Calc to run on the new ROMs. According to Mr. Waclo, a manufacturer has the option of either releasing two versions of his software, one for 1.0 and one for 1.1, or writing a universal version that will run on either ROM. The method chosen depends on the program itself. Disk users may have to read the fine print in ads to see if the program they want is written for the version of ROM they have. I am sure most manufacturers would want to create a single version for both ROMs if at all possible.

J&M Disk Controller Card

The 1.1 ROM was delivered to me with the J&M driver controller card I obtained from J&M Systems, LTD of Albuequerque, NM. I mentioned in my August column that I had seen these cards at Rainbowfest and recently ordered one. The card is enclosed in an aluminum case and the workmanship is excellent. It is available with or without the 1.1 ROM. With the ROM, it is completely compatible with any Radio Shack software. I have interfaced the card with the TEC drive that is standard on the CoCo, with an MPI drive, and with Tandon drives, and all seem to work fine.

Two of the nicest features about the J&M card are digital pre-compensation (there are no potentiometers to get out of adjustment), and gold-plated card contacts. One of the most troublesome areas of the Radio Shack card has been the lack of solid connections to the ROMport and drive-cable connector. The Radio Shack card needs constant cleaning to prevent oxidation from creating poor connections at the card ends. The J&M card should solve that problem. Surprisingly, the cards retail for only $149.95. When mated with a Tandon or Teac drive, the J&M system is a powerful package. J&M told me they will be releasing a 1.1 super ROM that will have all the features of the new RS ROM, but also allow you to change step rates and other disk parameters. The ROMs will probably be available when you read this.

Elite*Calc

A high-quality spread-sheet program is finally available for the CoCo. The program is very much like the Super Calc that runs on larger business computers. I am truly impressed with Elite*Calc. Though it has a few minor bugs, it provides nearly all of the functions that other commercial spread sheets make available. In addition, it has a few features others don't have. I have had the opportunity to work with the program since I purchased it at Rainbowfest and didn't realize it had IF-THEN-ELSE capacity until I read a recent ad. The one weak point of Calc is the manual. It is written for those who are somewhat familiar with a spread-sheet program. Use a book or tutorial for Super Calc or another spread sheet if you have problems understanding the Elite*Calc manual. Transferring examples to Elite*Calc syntax should not be difficult.

I am so impressed with the program that I have started a nationwide Elite*Calc User's Group. The major objective of the group is to provide a clearinghouse for spread sheets. People who have written sheets can exchange them for sheets others have written. This should create a large supply of useful routines, and allow people to modify

already existing routines, rather than reinvent the wheel by having to completely write their own.

Programmers' Utilities

Since this month's issue covers programming techniques, I wanted to mention several useful utilities available from Micrologic. E. R. Bailey of Micrologic has several programs available that process BASIC programs saved in ASCII format. The one I use most often is LLSTFM, a BASIC program formatter that provides a paged, titled, and dated program listing. The only nicety that could be added to the program is the inclusion of spaces between keywords in packed programs. Other utilities include a line-number cross referencer and a variable cross referencer. These programs scan your BASIC program and identify variables and line numbers and cross reference them for easy tracking of program logic.

A disk directory program does what RS DOS should: allows printed directories, a paged screen directory, and available space data. I will have more comments on these programs in a future column.

Another useful programmer's utility is the Platinum Worksaver by Platinum Software. I don't write any programs without first loading the worksaver. Single key keyword entry, redefinable keys, and a super screen editor make the program worth far more than the $35.00 it costs. A keyboard overlay is included that can be used to remind you of key definitions saved with each program. The worksaver has a small overhead of a couple of hundred bytes, but complete screen editing and the ability to integrate Worksaver features into a program more than make up for the small overhead.

MICRO

You may contact Mr. Steiner at 508 Fourth Ave. NW, Riverside, ND 58078.

Programming Techniques

by Phil Daley

What is a programming technique? There are probably as many different definitions as there are programmers. The *Computer Dictionary* defines "programming" as "the process of creating a program" and *Webster's Dictionary* defines "technique" as a "method of handling materials of an art". This provides us with a general idea of methods for creating a program.

There are several categories of such methods; e.g., improving program legibility or internal documentation, increasing the speed of the program, providing the program user with a crash-proof friendly environment, providing future program changers with a structured well-designed program to modify, and using features built into a particular computer to their best advantage. This issue of MICRO presents a collection of techniques for BASIC, Pascal, and machine-language programs to enhance your knowledge and enjoyment of your microcomputer.

Pascal users can look for a primer on gaming techniques in Pascal including a simple, but modifiable, example of a Pascal game. Also, there is a pseudo-random-number generator, usable for card dealing or anywhere random numbers are needed.

For machine-language buffs we have some techniques for faster matrix operations, an m-l string search, and an extremely sophisticated routine for animation with CoCo graphics. You can plot figures in between vertical scanlines for a smooth-looking graphics display.

BASIC fans can really make their programs more friendly with the first of a series of user-friendly techniques. This month the subject is the directory menu for easy selection of programs from your disk collection. Add this program to all your bootable disks. We also have programs for screen data plotting for all our covered micros. This is a prelude to our character-graphics coverage for November.

We think you will enjoy our new expanded coverage with listings for all micros of as many programs as possible.

MICRO

The Directory Menu

by Phil Daley

One of the most important and yet most difficult tasks for a beginning computer user is to perform a directory search of a disk and to load and run a program. While a sophisticated user would have no trouble determining which programs are runnable (or B-runnable), the beginner is confronted with all sorts of file-types and meaningless filenames and has to remember the correct syntax (I even get mixed up going from computer to computer) and proper commands to get the computer to accomplish the task at hand. It would be much easier to have a single command to learn or, in the case of auto-booting computers, no commands to learn at all.

Directory Menu is a user-friendly utility program that will display a directory of all programs on a disk and allow you to load and run any program at the touch of a key. It will be useful for beginners who want to simplify their disk organization and implement auto-running menu selection on their own disks. It will also be useful for programmers who want their programs to be the ultimate in user-friendly/simple menu selection.

Many people who buy computers do not want to learn about programming or memorize commands. Others are eager to begin writing their own programs. In either case, most people soon acquire a growing library of programs, storing many on a single disk. On auto-booting systems, such as the Atari and Apple, most people write DOS onto a disk before copying or saving other programs. This allows you to boot using any disk and to run a "HELLO" or menu program automatically. These procedures may seem quite simple to the experienced user, but the beginner can easily become confused and frustrated. Whether you are a beginner or not, you can save yourself time and effort by letting your computer do the work.

Wouldn't it be easier to just turn on the computer and have the machine tell you the names of each file on a disk, assign it a number, then let you simply type your selection by number? Wouldn't it be easier to let the computer figure out if your selection is a BASIC or machine-language program, then run it for you? If you answered yes to these questions, read on.

The program creates a MENU and displays a page of program names one page at a time. To load and run a program displayed, merely type the number of the program appearing to the left of the program name and press return. Versions are included for the Apple, Atari, Color Computer, and Commodore 64. The listings and descriptions of the individual programs appear in the appropriate system sections of the magazine.


MONSTER SMASH™
R.I.P.
Ross
by Dave Eisler
It's midnight... Do you know where your monsters are?
It's the dead of night. You're the master of the graveyard. The monsters are a wild and lively bunch who like to party and raise the dead. Can you use the Monster Smasher System and keep them where they belong ... six feet under? Only $29.95 for your Apple*, Commodore 64* or Atari*. Available at your computer dealer, or:
Copyright 1983 Datamost Inc.
8943 Fullbright Avenue, Chatsworth, CA 91311-2750 (213) 709-1202
VISA/MASTERCARD accepted $2.00 shipping/handling charge. (California residents add 6½% sales tax.)
Apple, Commodore 64, and Atari, are trademarks of Apple Computer Inc., Commodore Business Machines, and Atari Inc., respectively.

The author develops general-purpose plotting routines for the Commodore 64, VIC, PET, Atari 400/800/1200, TRS-80C, and Apple. Simple examples are used to demonstrate origins, scaling, rounding, and polar coordinates.

Plotting Data With Character Graphics

by Loren Wright

It is easy to plot mathematical functions or create attractive designs on your computer's character-graphics screen. This article will show you how, starting with simple concepts and simple functions and working up to a plot using polar coordinates. The result of these exercises will be a set of general-purpose routines that you can use to plot nearly any function on your computer screen. On the way, you'll learn about origins, scaling, rounding, symmetry, and coordinate systems.

The article is written for Commodore, Atari, and TRS-80C screens. Nearly all differences are included in the start-up modules listed on page 29. The Apple's screen memory is not continuous, so these routines will not apply directly to the Apple. However, I have provided a separate set of modifications that will allow you to try the examples presented using the Apple's low-resolution graphics screen. See the listing on page 33.

The first thing we need is a system to describe the position of a point on the screen conveniently. The easiest system to learn is called a rectangular coordinate system. With this system two imaginary perpendicular lines intersect somewhere on the screen, usually at the lower left corner. This intersection point is called the origin, and the two lines are the axes. Each axis has a scale used for measurement. Depending on the data you are plotting, the size of the scale will vary. The horizontal and vertical scales may be different, but in our first examples they will be the same. By convention, the horizontal axis is called the x-axis and the vertical axis is called the y-axis.

The position of a point is described by measuring the distance from the origin to a perpendicular line leading from the x-axis to the point, and by similarly measuring on the y-axis the distance from the origin to a perpendicular line leading to the point. These two measurements, one along the x-axis and the other along the y-axis, uniquely describe that position. In figure 1, point A has x-y coordinates of 4,4; point B is at 5,3. Negative coordinates are possible, but we will save them until later.

Start-up Modules

Commodore 64

```
10 OG=1024: CM=55296:
   XX=24:  MX=39: MY=24
20 POKE 53280,2: POKE 53281,1:
   CC=2
40 PRINT "[□]"
```

VIC-20

```
10 OG=4*(PEEK(36866)
   AND 128)+64*(PEEK(36869)
   AND 112)
11 CM=37888+4*(PEEK
   (36866)AND 128): MX=21:
   MY=22: XX=24
20 POKE36879,PEEK(36879)AND
   8OR(16*2+1): CC=2
40 PRINT "[□]"
```

PET/CBM

```
10 OG=32768: XX=24: MX=39:
   MY=24
40 PRINT "[□]"
```

Note: Don't use line 3050!

All Commodore Machines:

```
900 GET T$: IF T$=" " THEN 900
999 STOP
```

TRS-80C
(Requires Extended Color BASIC)

```
10 OG=1024: XX=88
20 MX=31: MY=15
30 CLS
.
.
.
900 IF INKEY$=" " THEN 900
999 STOP
```

Atari 400/800/1200

```
10 OG=256*PEEK(89)+
   PEEK(88)
20 XX=56:MX=39:MY=23
30 SETCOLOR 2,13,10:
   SETCOLOR1,0,0:
   SETCOLOR4,3,6
40 POKE752,1:PRINT "↖"
.
.
.
900 IF  PEEK(764)=255
    THEN 900
999 POKE 752,0:STOP
```

Your Computer Screen

Most home computers in their standard character modes have a memory-mapped screen. That is, each character position on the screen has a different memory location associated with it. Each memory location contains a number, which is a code representing the character in the corresponding position on the screen. The screen memory locations begin with the upper left corner of the screen and continue in sequence from left to right until the bottom right corner is reached. To put a character on the screen we just POKE the appropriate numerical code into the proper screen memory location. This isn't very handy, since the screen memory locations are big numbers. What we need is a system to convert from the handy x-y coordinate system described above to the computer's own memory-mapped system.

First you need to know a few things about your particular computer. The first lines of our program will contain information describing your computer's memory-mapped screen. Type in the start-up module lines listed for your computer. The □ in the Commodore listings and the ↖ in the Atari listing indicate the clearscreen character obtained by pressing SHIFT and CLR or CLEAR keys.

OG is the origin, or the address representing the upper left corner. MX is the number of columns less one, and MY is the number of rows less one. (For these demonstrations, the first column and first row are numbered 0.) XX contains the screen code for a capital X. The lines also contain statements that set screen, border, and character colors. The method varies with the computer.

Graphing a Function

Now that you have customized the program to your computer, add the following lines:

```
100 FOR X=0 TO MY: REM VIC use MX.
110 Y=X
120 X1=X: Y1=Y: GOSUB 3000
130 NEXT X

3000 REM X1=0 TO MX   Y1=0 TO MY
3030 PO=(MX+1)*Y1+X1
3040 POKE OG+PO,XX
3050 POKE CM+PO,0: REM
     VIC & C64 ONLY
3060 RETURN
```

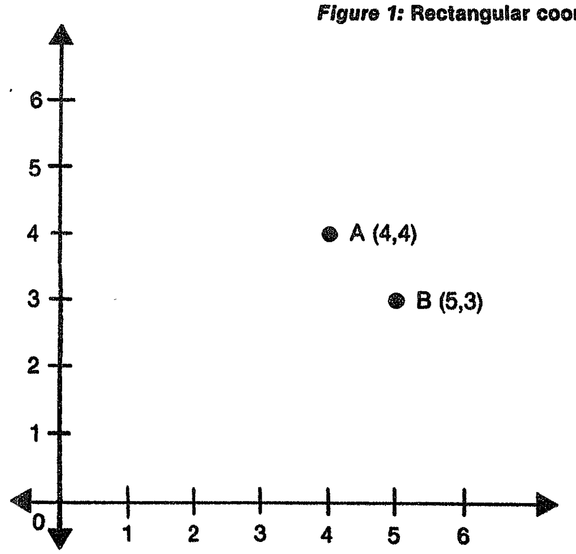
With any luck, you now have a line of X's leading down from the upper left corner. When we were discussing the coordinate system before, the origin was at the lower left, but this example makes the origin at the upper left. That's because the computer's screen memory starts with the upper left. We can correct this by adding line 3020 to the program:

```
3020 Y1=MY-Y1
```

Try RUNning the program again. This time the diagonal line starts at the

(Continued on page 31)

***Figure 1:* Rectangular coordinates**



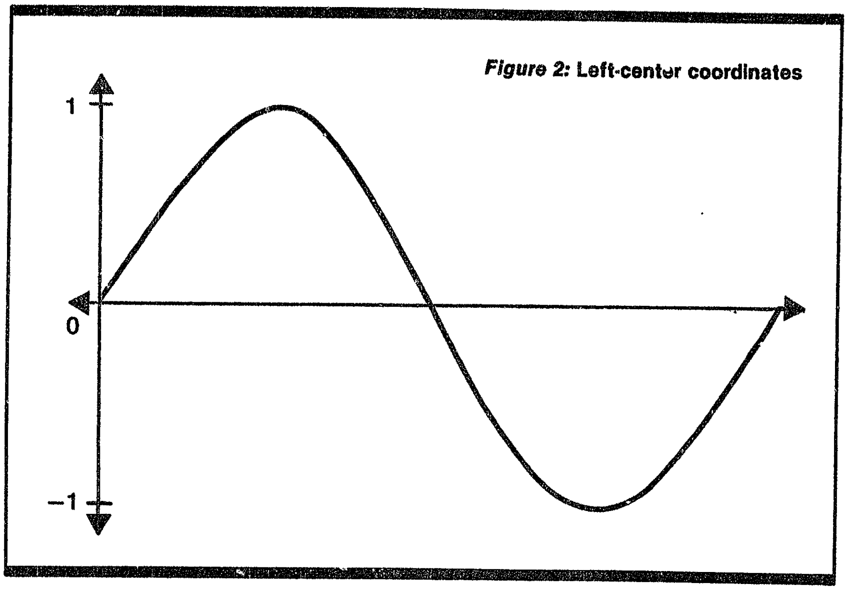


Figure 2: **Left-center coordinates**

bottom left. You have just graphed the mathematical function "Y=X" (see line 110 above) on your computer screen. Now try a new function. Make the changes in lines 100 and 110 so they appear as below:

```
100 FOR X=0 TO 10: REM CC use 7.
110 Y=2*X
120 X1=X: Y1=Y: GOSUB 3000
130 NEXT X
```

A new function "Y=2*X" will be plotted when you RUN the program.

Keeping It on the Screen

Note that line 100 had to be changed to prevent our graph from running off the screen. It is a good idea to be sure your points won't go off the screen since plotting out-of-range points could possibly crash your program. So far, I have been careful to choose values that will keep all points on the screen. You can't always be sure. By adding the following line to your plotting subroutine, you will avoid a disaster:

```
3010 IF (Y1 < 0)OR(Y1 > MY)OR
     (X1 < 0) OR(X1 > MX) THEN
     3060
```

This will cause the routine to return without plotting a point when either X or Y is out of range.

Rounding

Now let's try a new function. Substitute the following for lines 100-120:

```
100 FOR X=0 TO MX
110 Y=2*X/3
120 PRINT X;Y
```

When you RUN the program this time, a list of values instead of a graph will appear on the screen. Notice that most of the numbers calculated for Y are not whole numbers. Instead, many are decimals that end with a bunch of 3's or 6's. If you try to plot this on the screen by substituting 120 X1=X: Y1=Y: GOSUB 1000, you may or may not get a plot on the screen. The POKE command is designed to handle integers, not floating-point numbers, so your computer may give you an error message. Even if you were to use INT(Y), you would still get the wrong result. When X=2, Y is two-thirds. The INT( ) function turns two-thirds into 0, but we all know it should have been rounded up to 1.

Add the following line to your program:

```
70 DEF FNR(X)=INT(X+.5): REM Not
   for Atari!
```

This defines a general-purpose rounding function, which we can use with any floating-point variable anywhere in the program. (It is not restricted to the variable X, even though it was used in the definition.) Revise the main portion of your program so it appears as below:

```
100 FOR X=0 TO MX
110 Y=2*X/3
120 Y=FNR(Y): REM Atari use: 120
    Y=INT(Y+.5).
130 X1=X: Y1=Y: GOSUB 3000
140 NEXT X
```

RUN the program now and you will see that, although the line of X's is pretty ragged, at least the values for Y have been rounded correctly.

Scaling

The data you want to plot don't always fit neatly in the system we have set up. What happens when the numbers you calculate are larger than the maxima or when they are a lot smaller? The answer is to introduce a scaling factor. Consider the following example:

```
100 FOR X=0 TO 20
110 Y=X*X
120 X1=X: Y1=Y: GOSUB 3000
130 NEXT X
```

Be sure line 3010 is still in your plot subroutine and RUN the program. The first four or five points are plotted, but the remaining values are not. When X=5, Y=25, putting the point just off the top of the screen. When X=20, Y=400 and it would take a screen more than eight times as high to plot the point! To keep everything on the screen we need a scaling factor, which we will define in the set-up portion of the program:

```
90 SY=.05: REM CC use 90 SY=.037
```

Combine this with the rounding function to get a new line:

```
115 Y=FNR(Y*SY): REM Atari use: 115
    Y=INT(Y*SY+.5)
```

Each character unit in the vertical direction is now 20, but all the points will appear on the screen when you run the program. To determine the appropriate scaling factor, you must know the maximum value for your function. Divide the value of MY by the function maximum. This number is your scaling factor. To make things neat, round it down when it is less than one and up when it is greater than one. Twenty-two divided by 400 is .055, so I chose .05 for SY. (For CC 15/400=.0375, so use SY=.037.)

Now let's try an example that works the other way around. This time the values for Y will all fall between 0 and 1, so we'll need a large scaling factor. In addition we'll need a scaling factor for X. Delete lines 100-199 and type in the following program segment:

(Continued on next page)

```
90 PI = 3.14159265: SY = MY:
   SX = PI/MX
100 FOR X = 0 TO MX
110 X1 = SX*X
120 Y = SIN(X1)*SIN(X1)
130 Y = FNR(Y*SY):REM Atari use: 130
   Y = INT(Y*SY + .5)
140 X1 = X: Y1 = Y: GOSUB 3000
150 NEXT X
```

RUN the program first and you will see a rough bell curve plotted on the screen. Now for the details. Let's start with line 130. First Y (which I told you stays between 0 and 1) is multiplied by SY, which was set to MY in line 90. Then the rounding function is used. For the purposes of the calculation, X starts at 0 and goes up to π radians (the computer's SIN and other trigonometric functions use angles measured in radians rather than degrees). For the screen plot, we need integer values between 0 and MX. The solution is to keep X as an integer between 0 and MX (line 100) and use another variable X1 that increases in increments of 1/MX times π radians. X1 is used for the calculation in line 120, and subroutine 3000 gets X1, as usual. By the way, those of you with Commodore computers can eliminate the first statement in line 90 and replace the variable PI with the π character.

Origins

The previous example used the sine-squared function so we could keep the graph on the screen. Next we will graph a function that goes both positive and negative — a simple sine function. To do this, we need to move the origin. (Remember we placed it at the lower left corner of the screen at the very beginning.) The new origin is at the left edge in the middle (see figure 2). Delete lines 100-150 and replace them with the following program:

```
25 CY = INT(MY/2)
90 PI = 3.14159265:SY = INT(MY/2):
   SX = 2*PI/MX
100 FOR X1 = 0 TO MX
110 X = SX*X1
120 Y = SIN(X)
130 Y = FNR(Y*SY): REM Atari use
   Y = INT(Y*SY + .5)
140 Y1 = Y: GOSUB 2500
150 NEXT X1
```

In addition to the main program, we add a new plot-routine entry that will automatically convert the origin from an assumed left-center origin to the usual lower-left:

```
2500 REM LEFT-CENTER ORIGIN
     CONVERSION
2510 Y1 = Y1 + CY
```

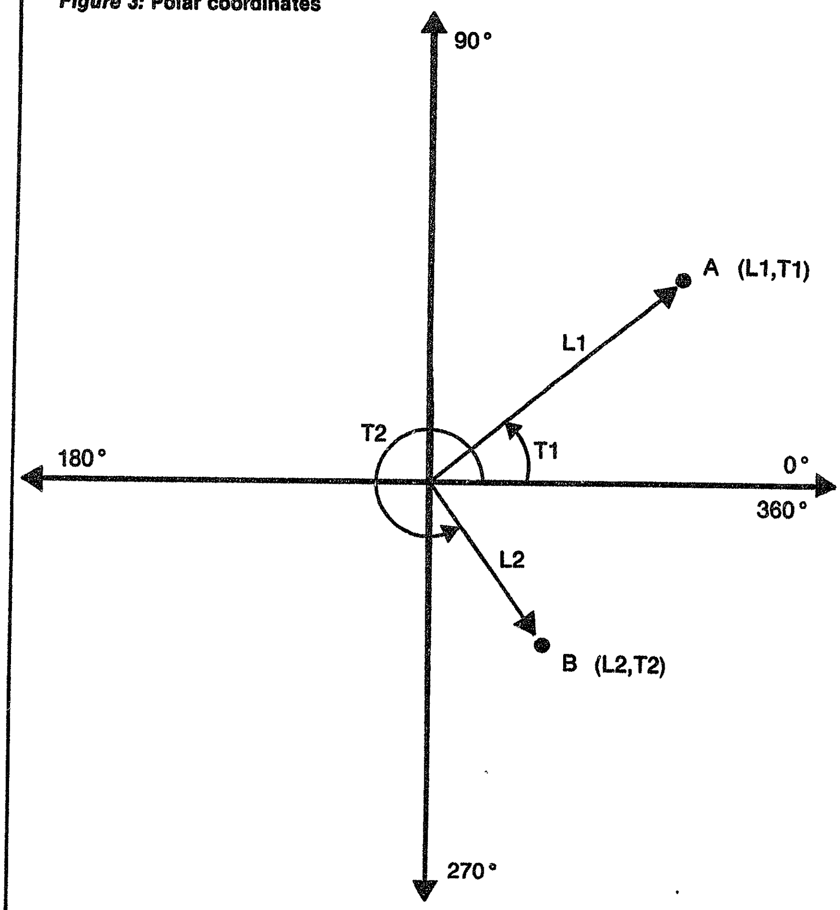
A lot has happened all at once. The sine function yields results between 1 and −1 or in a range of 2. To fill the screen vertically, we need a range of MY, so multiplying by INT(MY/2) converts the range to MY. As usual, the X value ranges in integer units from 0 to MX, so we use X1 as the FOR... NEXT loop index, and this gets passed directly to the plot routine. To calculate the Y values we use a separate variable X. By applying the horizontal scaling factor SX, we use X in the range 0 to 2 π in increments of 1/MXth of 2 π.

In the main program, we converted the range of the function to the full height of the screen. However, the low end of the range is −INT(MY/2). Adding CY (which conveniently is INT(MY/2)) makes the whole range positive, from 0 to MY. Instead of RETURNing to the main program, this routine automatically flows through to the plot routine we have been using all along. This way, you end up with a dual-purpose routine. If you are plotting with a lower-left origin, use GOSUB 3000; if you are plotting with a left-center origin, use GOSUB 2500.

Polar Coordinates

Most functions can be handled very well with the x-y coordinate system we have used so far. However, many functions, such as the circle we are about to plot, are best dealt with using a polar coordinate system. In this system, a point's position is described by its distance (in a straight line) from the origin and the angle this line makes with the horizontal in a counter-clockwise direction. See figure 3 for an illustration of this. Point A is described by the length of the line L1 and the

Figure 3: **Polar coordinates**



angle T1, while point B is represented by distance L2 and angle T2.

The computer handles angles in radians, while humans are more comfortable using degrees. Therefore we add a new function to convert angles in degrees to radians:

```
70 DEF FNR(X)=INT(X+.5):
   DEF FND(X)=X*PI/180
   (Atari use: 70 DEG)
```

To plot a circle, we need the origin in the center of the screen, so we need a center value for X and a new plot-routine entry to handle the center origin:

```
25 CY=INT(MY/2): CX=INT(MX/2)
2000 REM CENTER ORIGIN
     CONVERSION
2010 X1=X1+CX
```

This flows to the left-center entry and then to the main plot routine. Now you have a routine with three different entries, depending on the origin you are using. If you were using the routine with the x-y coordinate system and a center origin, then you would call the routine with GOSUB 2000. However, we are going to be using the polar coordinate system, so there will be a polar-to-rectangular section preceding it:

```
1000 REM POLAR (L,T) TO
     RECTANGULAR (X1,Y1)
     CONVERSION
1010 T1=FND(T): REM Atari use: T1=T
1020 X1=L*COS(T1)
1030 Y1=L*SIN(T1)
```

Since we are now performing so many adjustments on the data, it is best to save the rounding until just before the actual plotting takes place:

```
3025 X1=FNR(X1): Y1=FNR(Y1)
     (Atari: 3025 X1=INT(X1+.5):
     Y1=INT(Y1+.5)
```

Now for the main program. With the polar coordinate system, L will be the same for all points, and the angle will change in even increments over the full 360 degrees:

```
100 FOR T=0 TO 357 STEP 3
110 L=CY: REM VIC and Apple lo-res
    use: 110 L=CX
120 GOSUB 1000
130 NEXT T
```

Symmetry

Chances are that you won't get a circle, even if you thought you were plotting one. Your circle is probably more oval-shaped. This is because the space a character occupies is not perfectly square. Keep the circle plot on the screen and measure it with a ruler vertically and horizontally. Divide the vertical distance by the horizontal distance to get your symmetry factor. My factor came out to 1.33. By adding one line to your program, you can make the circle, and any other radial patterns, appear more symmetrical:

```
2005 X1=X1*1.33
```

Another easy pattern is a spiral:

```
100 FOR T=0 TO 717 STEP 3
110 L=CY*T/720
120 GOSUB 1000
130 NEXT T
```

By consulting geometry and trigonometry books, you will find other radial patterns, such as clover leaves, that can be easily programmed with the polar coordinate system.

Commodore Notes

In the start-up module we defined a variable CC, which we haven't used. We can use the variable in line 3050:

```
3050 POKE CM+ PO,CC: REM
     COMMODORE ONLY
```

RUNning any of the program examples will now cause the plot to appear in red instead of black. You may vary the color as each point is plotted by changing the value of CC, or you can set it once at the beginning.

As part of next month's character-graphics feature, I will present routines that use the Commodore machines' powerful graphics characters to achieve higher resolution.

Plotting Your Own Functions

You should be able to plot nearly any function using the general-purpose routines presented. Follow this simple procedure:

1. Type in the start-up lines and lines 900-999 as shown for your computer.
2. Type in the plotting subroutines. Store this on tape or disk. It will be your template for any plotting you will do. Lines 100-899 will vary from plot to plot; the others will stay the same.
3. Determine the origin. If X and Y both stay positive, use the lower-left origin (line 3000). If Y goes both negative and positive, use the left-center origin (line 2500). If both X and Y go both positive and negative, then the center origin should be used (line 2000). Radial patterns, such as circles, ellipses, spirals, and florettes, should use polar coordinates as well (line 1000).
4. Determine the horizontal range (X direction) of your plot. Usually you will want to cover the whole range in integer increments. Use this range to pass (in variable X1) to the plot subroutine and calculate another X value, using a horizontal scaling factor, for your Y value calculation. The horizontal scaling factor is determined by dividing the actual horizontal maximum by MX (or CX for center plots). (See the sine wave example.)
5. Determine the vertical range (Y direction) of your plot. Divide MY (or INT(MY/2) for left-center or center origins) by the actual Y range of the function to get the vertical scaling factor.
6. The rest is simple. Use a FOR...NEXT loop with X1 or X as the index. Remember to pass X1 and Y1 to the subroutine.

MICRO

Apple LO-RES Listing

(Substitute the following for the start-up module and subroutine 3000. Everything else should work as described in the text.)

```
10 HOME: POKE -16302,0: POKE
   -16304,0: CALL -1992
20 COLOR=15: MX=39: MY=47
.
.
.
900 IF PEEK(-16384) < 128
    THEN 900
910 TEXT: HOME
999 STOP
.
.
.
3000 REM X1=0 TO MX Y1=0
     TO MY
3010 IF (Y < 0) OR (Y1>MY) OR
     (X < 0) OR (X1>MX) THEN
     3060
3020 Y1=MY-Y1
3030 PLOT X1,Y1
3060 RETURN
```

STRUCTURED GAME DESIGN

Game writing in any language can be a rewarding experience. When writing games imagination and fantasy are transformed into graphic or text display, which requires a foundation in program design.

by Michael Allen

This article introduces the novice programmer to game programs. The Pascal language is used to provide clear examples. Advanced programmers will want to utilize these same concepts as assembly-language functions and procedures for greater speed in fast-paced games.

I have presented the examples as a compilable program so that the beginning game programmer is presented with a core around which to write his play screens. Any of the routines presented can be used as is or modified further for specific applications. Many of the routines can be used by more than one calling routine by adding parameters to the function headings.

When checking the list of variables, you will find that the majority used during the game are declared as global. The reason for this is to avoid using parameters, thus making it easy to keep track of the procedure calls during development.

In the fully compilable program that follows, the programmer needs only to fill in the screens. Using the procedures provided, a very playable game can be written (such as the sample screen). With a little effort and some modifications (try some graphics) this program can become the basic framework for any kind of game you might wish to write in Pascal.

```
{$S+}     {SWAPPING OPTION FOR COMPILING LARGE PROGRAMS}

PROGRAM  GAME;


USES APPLESTUFF; {NEEDED IN APPLE PASCAL
                  RANDOM, RANDOMIZE AND KEYPRESS ARE HIGHLY
                  USEFUL ROUTINES CONTAINED IN THIS UNIT}


CONST
  SIZE = 5;
TYPE
  INDEX = INTEGER;
VAR
  HI,POINTS : INTEGER[10];
  LEVEL,MEN : INDEX;
  LETTER : CHAR;
  WORDARR : PACKED ARRAY[1..SIZE] OF CHAR;
  COUNT,I : INDEX; {COUNT VARS-USE IN LOOPS AS NECCESSARY}
  HISCORE : TEXT;  {FILE FOR STORAGE OF HIGH SCORE VALUE}


PROCEDURE SCREEN; FORWARD; {TO AVOID UNDECLARED NAME ERRORS}
```

Sample Game requires: Pascal

The function 'COMPARE' compares a keyboard input to an array and returns a boolean value. This function would be useful in Hangman-type word games. The input variable 'LETTER' is compared to the array 'WORDARR' to check for a match.

```
FUNCTION COMPARE : BOOLEAN; {COMPARE AN INPUT TO A ARRAY}
VAR I : INTEGER;
BEGIN
  FOR I := 1 TO SIZE DO
  IF LETTER = WORDARR[I]
    THEN
    BEGIN
      WORDARR[I] := '0';
      COMPARE := TRUE;
      EXIT(COMPARE);
    END
  ELSE COMPARE := FALSE;
END;
```

The PLAYER UPDATE procedure is one of those routines that every game program must have. This procedure keeps the player informed as to the number of men left in play and the current score.

```
PROCEDURE PLAYERUPDATE; {UPDATE PLAYER INFO}
BEGIN
  WRITELN('POINTS = ',POINTS);
  IF MEN = 1 THEN WRITELN(MEN,' MAN LEFT')
    ELSE WRITELN(MEN,' MEN LEFT');
END;
```

Another necessary procedure is the 'END GAME' routine that tidies up the loose ends, such as storing the high score and displaying the player's level of achievement.

```
PROCEDURE ENDGAME;
BEGIN
  PAGE(OUTPUT);
  WRITELN('HIGH SCORE = ',HI);
  WRITELN;WRITELN;
  PLAYERUPDATE;
  GOTOXY(12,12);
  IF POINTS > 1000000 THEN WRITELN('SUPER MAN')
    ELSE IF POINTS > 500000 THEN WRITELN('ABOVE AVERAGE')
    ELSE IF POINTS > 250000 THEN WRITELN('COMMON MAN')
    ELSE IF POINTS > 100000 THEN WRITELN('ALMOST MAN')
    ELSE WRITELN('YOU COULD LOUSE UP A BRICK WALL');
    {RESET(HISCORE);
      IF POINTS > HI THEN WRITELN(HISCORE,POINTS);}
        {CAN BE USED AFTER CREATING DISK FILE}
  EXIT(PROGRAM);
END;


FUNCTION RAND : INTEGER; {ONE IN TEN RANDOM NUMBER}
CONST
  LOW = 1;
  HIGH = 100;
VAR
  MX,C,D : INTEGER;
BEGIN
  C := HIGH - LOW + 1;
  MX := (MAXINT - HIGH + LOW) DIV C + 1;
  MX := MX * (HIGH - LOW) + (MX - 1);
  REPEAT
    D := RANDOM ;
  UNTIL D < MX;
  RAND := LOW + D MOD C;
END;
```

The procedure DELAY is a variable delay. The length of delay is contingent upon the level of play. 'DELAY' is a simple loop repeated 'N' times, 'N' being dependent on the value of 'LEVEL'.

```
PROCEDURE DELAY; {DELAY BASED ON LEVEL OF PLAY}
VAR I : INTEGER;
BEGIN
  I := 2000 - 200 * LEVEL;
  IF I <600 THEN I := 600;
  REPEAT
    I := I - 1;
  UNTIL I = 0;
END;
```

The HALLMON (hall monitor) procedure is a procedure for computing the chance (50/50 in this application) of escaping or avoiding an obstacle. The obstacle can be mobile or stationary. This procedure can also be used in combination with 'PERCENTILE', 'CHANCE', and 'FUMBLE' for greater flexibility of action with multiple branching.

```
PROCEDURE HALLMON; {WANDERING OBSTACLE}
VAR I : INTEGER;
BEGIN
  I := RAND;
  GOTOXY(12,14);
  WRITELN(' ':80);
  GOTOXY(12,14);
  IF ODD(I)
    THEN
    BEGIN
      POINTS := POINTS + 1000 * LEVEL;
      WRITELN('YOU GOT IT');
    END
  ELSE
  BEGIN
    MEN := MEN - 1;
    WRITELN('IT GOT YOU!');
    EXIT(SCREEN);
  END;
END;
```

The function CHANCE is simply a 1 - 100 random-number generator.

```
FUNCTION CHANCE : INTEGER; {ONE IN A HUNDRED RANDOM NUMBER}
CONST
  LOW = 1;
  HIGH = 100;
```

```
VAR
  MX,C,D : INTEGER;
BEGIN
  C := HIGH - LOW + 1;
  MX := (MAXINT - HIGH + LOW) DIV C + 1;
  MX := MX * (HIGH - LOW) + (MX - 1);
  REPEAT
    D := RANDOM ;
  UNTIL D < MX;
  CHANCE := LOW + D MOD C;
END;


PROCEDURE LINE1; {RANDOM TEXT GENERATION}
BEGIN
  RANDOMIZE;
  CASE RAND OF
    1,6 : WRITELN('QUIET AS A MOUSE');
    2,7 : WRITELN('A CAT WOULD BE JEALOUS');
    3,8 : WRITELN('EVER CONSIDER A DISHONEST PROFFESSION');
    4,9 : WRITELN('THE NEXT ONE WON'T BE SO EASY');
    5,10: WRITELN('YOU'VE  DONE THIS BEFORE RIGHT !!');
  END;
END;


PROCEDURE LINE2;
BEGIN
  RANDOMIZE;
  CASE RAND OF
    1,6 : WRITELN('THAT WAS A CLOSE ONE');
    2,7 : WRITELN('BE CAREFUL !!');
    3,8 : WRITELN('YOU'RE  A REAL KLUTZ TODAY');
    4,9 : WRITELN('EVER CONSIDER A ''QUIET'' HOBBY ??');
    5,10: WRITELN('YOU ALMOST BLEW IT ');
  END;
END;
```

The procedure FUMBLE decides whether or not the player has fumbled (1 - 64); if he has he loses one man. If the player has not fumbled 'LINE2' is called.

```
PROCEDURE FUMBLE;
VAR CH : INTEGER;
BEGIN
  CH := CHANCE;
  IF CH >= 65 THEN LINE2
    ELSE BEGIN
           WRITELN('YOU WOKE THE WHOLE NEIGHBORHOOD');
           MEN := MEN - 1;
           EXIT(SCREEN);
         END;
END;
```

The PERCENTILE procedure uses the number generated by 'CHANCE' to make decisions — in this case to decide between 'LINE1' (76 - 100), 'LINE2' (26 - 75), or 'FUMBLE' (1 - 25). 'LINE1' and 'LINE2' simply print text messages selected at random.

```
PROCEDURE PERCENTILE;
VAR CH : INTEGER;
BEGIN
  CH := CHANCE;
  IF CH > 75 THEN LINE1
    ELSE IF CH > 25 THEN LINE2
    ELSE IF CH >= 1 THEN FUMBLE;
END;
```

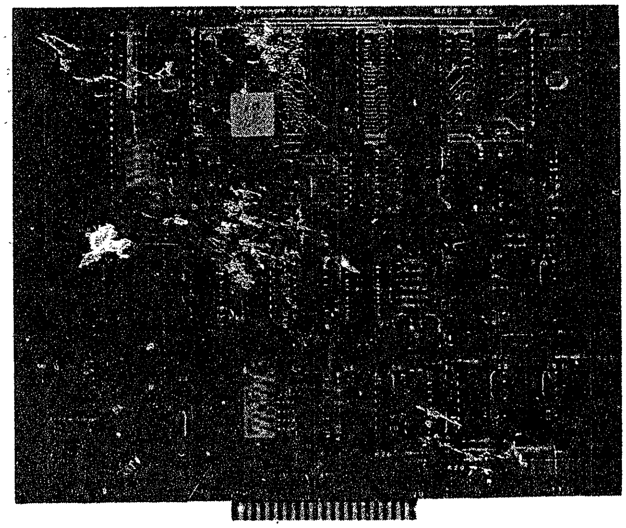
The INDAT procedure is the same as performing a 'READ' statement except now allowable inputs are selected by the programmer. In the example, only 'A' through 'Z' will be accepted.

```
PROCEDURE INDAT;
BEGIN
  GET(KEYBOARD);
  WHILE NOT (KEYBOARD^IN ['A'..'Z']) DO
    GET(KEYBOARD);
  LETTER := KEYBOARD^;
END;


PROCEDURE SCREEN; {SAMPLE SCREEN}
VAR
  LIMIT : INDEX;
  CORRECT : INDEX;
BEGIN
  WORDARR := 'GAMES';
  CORRECT := 0;    {0 CORRECT GUESSES}
  LIMIT := 20;     {20 GUESSES}
  WRITELN('YOU HAVE TWENTY GUESSES TO FIND THE LETTERS');
  WRITELN('IN A FIVE LETTER WORD');
  WRITELN;
  WRITELN('WHEN YOU FIND THE LETTERS UNSCRAMBLE THEM');
  WRITELN('TO FIND THE WORD');
  WRITELN;
  WRITELN('START GUESSING. IF YOUR GUESS IS CORRRECT');
  WRITELN('THE COMPUTER WILL PRINT THE LETTER ON THE SCREEN.');
  REPEAT
    INDAT;
    IF COMPARE THEN
                BEGIN
                  WRITE(LETTER);
                  CORRECT := CORRECT + 1;
                END;
    LIMIT := LIMIT - 1;
  UNTIL (CORRECT = 5) OR (LIMIT = 0);
  WRITELN;WRITELN;
  IF LIMIT = 0 THEN
                BEGIN
                  WRITELN('OUT OF GUESSES');
                  MEN := MEN - 1;
                END;
  POINTS := (2000 - (100 * (20 - LIMIT))) * LEVEL;
  PLAYERUPDATE;
END;


PROCEDURE SCREEN2;
BEGIN
END;


PROCEDURE SCREEN3;
BEGIN
END;


PROCEDURE SCREEN4;
BEGIN
END;


BEGIN {MAIN}
  {RESET(HISCORE,'HISCORE');
  {FILE ON DISK FOR SAVING LAST HIGH SCORE
    READLN(HISCORE,HI);}
  MEN := 6;
  POINTS := 0;
  LEVEL := 1;
  REPEAT
    SCREEN;
    IF MEN = 0 THEN ENDGAME;
    SCREEN2;
    IF MEN = 0 THEN ENDGAME;
    SCREEN3;
    IF MEN = 0 THEN ENDGAME;
    SCREEN4;
    LEVEL := LEVEL + 1;
  UNTIL (MEN = 0);
  ENDGAME;
  {CLOSE(HISCORE,LOCK);}
END.
```

MICRO

Michael Allen has been programming in Pascal for about two years. He is currently employed at Martin Marietta Aerospace as a quality engineer and is working toward a B.S. in Computer Science at Chapman College. You may contact Mr. Allen at 1500 West Cherry, Lompoc, CA 93436.

VIDEO TERMINAL BOARD 82-018



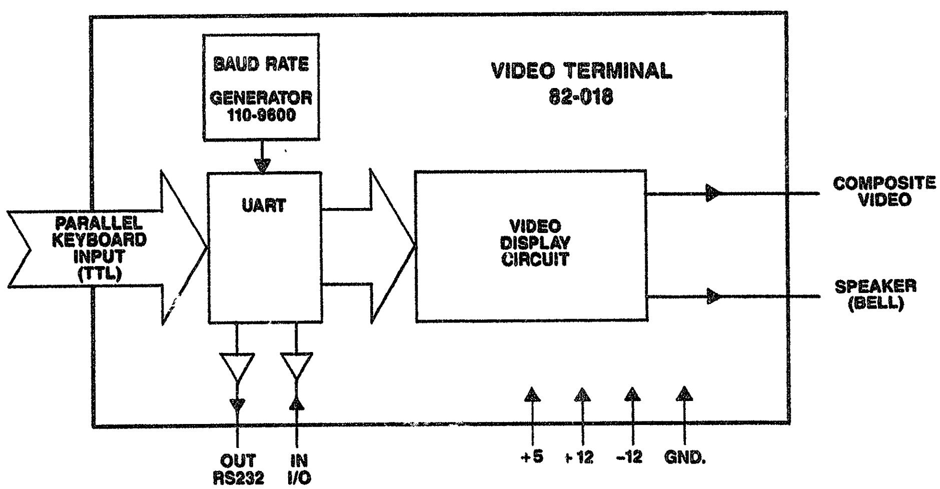
This is a complete stand alone Video Terminal board. All that is needed besides this board is a parallel ASCII keyboard, standard NTSC monitor, and a power supply. It displays 80 columns by 25 lines of UPPER and lower case characters. Data is transferred by RS232 at rates of 110 baud to 9600 baud — switch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

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

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

Features

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



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

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

Both versions come with complete documentation.

JOHN BELL ENGINEERING, INC.

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

SEND $1.00 FOR CATALOG

(415) 592-8411
WILL CALL HOURS: 9am - 4pm

MasterCard VISA

#249

Rapid String/ Substring Search

This Applesoft BASIC program demonstrates the use of a machine-language program to obtain a rapid array search. String-array variables can be quickly scanned for string/substring variables making it possible to search up to 11000 characters in a fraction of a second.

by L. S. Reich

When an Applesoft BASIC program is employed to search an array containing hundreds or thousands of names and addresses, the results can be relatively slow in forthcoming. This is a major reason that assembly language (ASL) is used for the tasks.

The ASL Program

Listing 1 is a 6502 assembly-language program that utilizes two important Applesoft interpreter functions — PTRGET ($DFE3) and GETARYPT ($F7D9). PTRGET allows the attainment of descriptor addresses for string variables using VARPNT ($83,84) locations. The descriptor consists of three bytes, the first leading to the string length and the next two to the string address (low/high bytes, respectively). The manner in which the string/substring (B$) length is obtained, as well as the pointers to the addresses containing the string variable values, is depicted in lines 27-40.

The utilization of GETARYPT allows the attainment of descriptor addresses for variable string arrays. Initially, GETARYPT is employed to find the location of a string array header whose address is stored in LOWTR ($9B,9C) locations. Seven bytes are then added to find the location of the descriptor for the first array element (A$(0)), and the next descriptor is three bytes away, etc.; e.g., lines 99-105 (also see p. 137 of the *Applesoft II BASIC Programming Reference Manual).* This is depicted in lines 43-50. In lines 51-59, the string array length is obtained as well as array pointers to addresses containing the variable string array values. In lines 61-67, the string array length for A$(J) is compared with the string length for B$. If the latter value is larger, then another descriptor is obtained for the next string array, lines 99-106; otherwise, the difference in the lengths is stored in PTR+1 to determine the last test position. In lines 69-86, a counter is used (PTR) to determine whether the last test position has been reached; if not, then further comparisons are made between A$(J) and B$.

A successful comparison (GOTWRD) results in location $1A possessing a value of $FF (lines 108-110) and a return to the BASIC program. Lines 83-86 allow A$(J) to be compared along its length with string/substring B$. In lines 87-97, the values in counters $1B,1D (low/high bytes, respectively) are compared with the number-of-words limit established in locations $3AA,3AB (low/high bytes, respectively). This limit was imposed in the associated BASIC program. When this limit (N in the BASIC program) is attained, BASIC is reentered (line 111).

The BASIC Program

Listing 2 is the BASIC program. As previously mentioned, the number-of-words limit (N) is stored in locations $3AA,3AB (lines 40 and 70). If the RETURN key is used for the key word (B$), then the program ends (lines 90-100). In line 110, if D$ = "N" then only the first string array (A$(J)) that shows a match with the string/substring (B$) will be displayed. However, if D$ = "Y" then all string arrays containing B$ will be displayed.

Line 310 represents the first string array, A$(0). In line 130, the ASL program is called using CALL 768,B$,A$. The ampersand function may be used instead of "CALL 768" by appropriate modification of the program. This change should result in a slightly faster program. When a successful match has been obtained [($1A) = $FF], the matched string variable is displayed (line 150). If D$ = "N" the program then ends; otherwise ($1A) is reset to zero and the matching is continued (lines 170-180).

References

1. G. B. Little, MICRO, (57:32).
2. L. Reynolds, *Call-A.P.P.L.E.*, p. 26 (January 1981).
3. B. Sander-Cederlof, *Apple Assembly Line*, #7, p. 18 (1981).
4. J. Crossley, *Call-A.P.P.L.E. in Depth*, #1, p. 51 (1981).
5. C. Kluepfel, *Call-A.P.P.L.E.*, p. 50 (May 1981).

String Search
requires:
Apple II

Listing 1

```
                         *                                   *
                         *      Rapid String Search          *
                         *                                   *
                         *      By L.S.  Reich               *
                         *                                   *
                         *      CALL 768,B$,A$               *
                         *                                   *
                         *   Search for B$ in A$ using ML    *
                         *   Routine from Applesoft BASIC    *
                         *                                   *
                         *      Copyright (C) 1983           *
                         *       By MICRO Ink                *
                         *      10 Northern Blvd.            *
                         *      Amherst, NH 03031            *
                         *                                   *
                         ************************************

                                  START
                         ** APPLESOFT POINTERS:
                         PTR      GEQU   $10
                         VARPNT   GEQU   $83
                         LOWTR    GEQU   $9B
                         LENB     GEQU   $D0
                         CHKCOM   EQU    $DEBE
                         PTRGET   EQU    $DFE3
                         GETARYPT EQU    $F7D9

                                  ORG    $300

0033  0300 20BEDE                 JSR    CHKCOM
0034  0303 20E3DF                 JSR    PTRGET
0035  0306 A000                   LDY    #$00
0036  0308 841A                   STY    $1A          Initialize
0037  030A 841B                   STY    $1B
0038  030C 8410                   STY    PTR
0039  030E 841D                   STY    $1D
0040  0310 B183                   LDA    (VARPNT),Y
0041  0312 85D0                   STA    LENB
0042  0314 C8                     INY
0043  0315 B183                   LDA    (VARPNT),Y Get Pointers
0044  0317 8506                   STA    $06          and Store
0045  0319 C8                     INY
0046  031A B183                   LDA    (VARPNT),Y
0047  031C 8507                   STA    $07

0049  031E 20BEDE                 JSR    CHKCOM
0050  0321 20D9F7                 JSR    GETARYPT
0051  0324 A59B                   LDA    LOWTR
0052  0326 18                     CLC
0053  0327 6907                   ADC    #$07         Get to first
0054  0329 85E7                   STA    $E7          Array Variable
0055  032B A59C                   LDA    LOWTR+1
0056  032D 6900                   ADC    #$00
0057  032F 85E8                   STA    $E8
0058  0331 A000          LOOP     LDY    #$00
0059  0333 B1E7                   LDA    ($E7),Y
0060  0335 85D1                   STA    LENB+1
0061  0337 C8                     INY
0062  0338 B1E7                   LDA    ($E7),Y      Get Array
0063  033A 8508                   STA    $08          Pointers and
0064  033C C8                     INY                 Store
0065  033D B1E7                   LDA    ($E7),Y
0066  033F 8509                   STA    $09

0068  0341 A5D1                   LDA    LENB+1
0069  0343 C5D0                   CMP    LENB
0070  0345 902A                   BLT    AGAIN
0071  0347 A5D1                   LDA    LENB+1
0072  0349 38                     SEC
0073  034A E5D0                   SBC    LENB
0074  034C 8511                   STA    PTR+1
0075  034E A900                   LDA    #$00
0076  0350 8510                   STA    PTR
```

(Continued on next page)

Listing 1 *(continued)*

```
0352 A0FF     Y1        LDY   #$FF
0354 C8       Y2        INY
0355 F004               BEQ   Y4
0357 C4D0               CPY   LENB       Finished?
0359 F03D               BEQ   GOTWORD    Yes
035B B108     Y4        LDA   ($08),Y    No
035D D106               CMP   ($06),Y
035F F0F3               BEQ   Y2
0361 E610               INC   PTR
0363 A511               LDA   PTR+1
0365 C510               CMP   PTR
0367 9008               BLT   AGAIN
0369 E608               INC   $08
036B D0E5               BNE   Y1
036D E609               INC   $09
036F D0E1               BNE   Y1
0371 A000     AGAIN     LDY   #$00
0373 8410               STY   PTR
0375 E61B               INC   $1B        Number Words
0377 D002               BNE   Y3          Lobyte
0379 E61D               INC   $1D        Number Words
037B A51B     Y3        LDA   $1B         Hibyte
037D 38                 SEC              Words Limit
037E EDAA03             SBC   $3AA        Lobyte
0381 A51D               LDA   $1D        Words Limit
0383 EDAB03             SBC   $3AB        Hibyte
0386 1015               BPL   OVER       Finished

0388 A5E7               LDA   $E7        Get new
038A 18                 CLC              Descriptor
038B 6903               ADC   #$03       For next array
038D 85E7               STA   $E7
038F A5E8               LDA   $E8
0391 6900               ADC   #$00
0393 85E8               STA   $E8
0395 4C3103             JMP   LOOP

0398 A9FF     GOTWORD   LDA   #$FF       Sucess
039A 851A               STA   $1A        Marker
039C 60                 RTS
039D 4C03E0   OVER      JMP   $E003      BASIC
03A0                    END
```

Listing 2

```
10  REM **APPLESOFT PROGRAM FOR USE WITH  ML SEARCH ROUTINE**
15  TEXT : HOME : VTAB 10: PRINT "   SETTING UP STRING SEARCH"
20  IF  PEEK (927) = 224 THEN 40
30  PRINT  CHR$ (4)"BLOAD STRING.OBJ,A$300"
40 N = 257: REM NUMBER OF WORDS SEARCHED
50  DIM A$(N)
60 HI =  INT (N / 256):LO = N - HI * 256
70  POKE 938,LO: POKE 939,HI: REM NUMBER OF WORDS LIMIT
80  FOR I = 0 TO N - 1: READ A$(I): NEXT
90  INPUT "GIVE KEY WORD: ";B$
100  IF B$ = "" THEN 200
110  INPUT "ALL OR FIRST OCCURANCE (A/F)? ";F$
120  REM ******************************************
130  CALL 768,B$,A$: REM GET A$'S MATCHING B$'S
140  REM ******************************************
150  IF  PEEK (26) = 255 THEN  PRINT A$( PEEK (29) * 256 +
     PEEK (27)): PRINT :REM DISPLAY SIGNAL
160  IF F$ = "F" THEN 200
170  POKE 26,0: REM REINITIALIZE
180  CALL 881: REM CONTINUE MATCHING
190  GOTO 150
200  END
300  DATA LEO,LEON,SAL,DORIS,MARGE,BILL,JOHN,WALTER,
        PETER,HANS,GEORGE,CHARLES,FRANK,KELLY,RORY,
        TED,LEONARD,JACK,DAVID,WILLIAM,WILLA,MARY,
        MARGO,ALICE,JASON,MARK
```

(continued)

Listing 2 *(continued)*

```
310  DATA GRAYSON,ABEL,BETTY,CARA,DALE,ELLA,FRANCIS,
          FRANCES,GAIL,HARRY,HAROLD,ERNEST,JILL,KILMER,
          NORMAN,OLIVER,PHILLIP,ROBERT,SALLY,THOMAS,
          ARNOLD,BUELL,MAYNARD,MOIRA,HALEY
320  DATA MERLE,MURIEL,JACKSON,HILLARY,HOLMES,STANLEY,
          MARGARET,SIMPSON,SYLVIA,BERNICE,BERNARD,CARRIE,
          CHARLOTTE,PEARL,MINNIE,NORMA,ANN,VIRGINIA,GRACE,
          ROSE,ROSEMARY,LILLIAN,LOUISE,LILA,NELLIE
330  DATA LEWIS,LEWISON,LENNY,HERMAN,LESTER,LEMUEL,SAMUEL,
          HOMER,LARS,WOLF,XAVIER,YOUNG,ZELMO,ELMER,HILMER,
          BARRY,STEVE,STEPHEN,PHILO,GARRY,PRESTON,SANDRA,
          SANDY,ALBERT,CARMEN,ALDO,ZERO
340  DATA LEWIS,LEWISON,LENNY,HERMAN,LESTER,LEMUEL,SAMUEL,
          HOMER,LARS,WOLF,XAVIER,YOUNG,ZELMO,ELMER,HILMER,
          BARRY,STEVE,STEPHEN,PHILO,GARRY,PRESTON,SANDRA,
          SANDY,ALBERT,CARMEN,ALDO,ZERO
350  DATA MERLE,MURIEL,JACKSON,HILLARY,HOLMES,STANLEY,
          MARGARET,SIMPSON,SYLVIA,BERNICE,BERNARD,CARRIE,
          CHARLOTTE,PEARL,MINNIE,NORMA,ANN,VIRGINIA,GRACE,
          ROSE,ROSEMARY,LILLIAN,LOUISE,LILA,NELLIE
360  DATA GRAYSON,ABEL,BETTY,CARA,DALE,ELLA,FRANCIS,FRANCES,
          GAIL,HARRY,HAROLD,ERNEST,JILL,KILMER,NORMAN,
          OLIVER,PHILLIP,ROBERT,SALLY,THOMAS,ARNOLD,BUELL,
          MAYNARD,MOIRA,HALEY
370  DATA LEO,LEON,SAL,DORIS,MARGE,BILL,JOHN,WALTER,PETER,
          HANS,GEORGE,CHARLES,FRANK,KELLY,RORY,TED,LEONARD,
          JACK,DAVID,WILLIAM,WILLA,MARY,MARGO,ALICE,JASON,MARK
380  DATA GRAYSON,ABEL,BETTY,CARA,DALE,ELLA,FRANCIS,FRANCES,
          GAIL,HARRY,HAROLD,ERNEST,JILL,KILMER,NORMAN,OLIVER,
          PHILLIP,ROBERT,SALLY,THOMAS,ARNOLD,BUELL,MAYNARD,
          MOIRA,HALEY
390  DATA MERLE,MURIEL,JACKSON,HILLARY,HOLMES,STANLEY,MARGARET,
          SIMPSON,SYLVIA,BERNICE,BERNARD,CARRIE,CHARLOTTE,
          PEARL,MINNIE,NORMA,ANN,VIRGINIA,GRACE,ROSE,ROSEMARY,
          LILLIAN,LOUISE,LILA,NELLIE
400  DATA LEWIS,LEWISON,LENNY,HERMAN,LESTER,LEMUEL,SAMUEL,
          HOMER,LARS,WOLF,XAVIER,YOUNG,ZELMO,ELMER,HILMER,
          BARRY,STEVE,STEPHEN,PHILO,GARRY,PRESTON,SANDRA,SANDY,
          ALBERT,CARMEN,ALDO,ZERO
```

AICRO™

Directory Menu for the Apple

by Keith Davison and Phil Daley

The Apple menu program uses a machine-language routine to convert the normal CATALOG routine into a string array of the individual entries. CAT.OBJ (listing 1) is the assembly-language listing of the routine to accomplish this task. If you don't have an assembler, enter the monitor with a CALL-151 and type

Listing 1

```
                  *************************************
                  *                                   *
                  *         MENU DIRECTORY            *
                  *         BY KEITH DAVISON          *
                  *                                   *
                  *************************************

                  CSW     GEQU   $36
                  INDEX   GEQU   $5E
                  RTS     EQU    $60
                  FRESPC  GEQU   $71
                  VARPNT  GEQU   $83
                  RETURN  EQU    $8D
                  TXTPTR  GEQU   $B8
                  COUNT   GEQU   $FE
                  FLAG    GEQU   $FF
                  KEYBUF  EQU    $200
                  SETPTR  EQU    $3EA
                  PAUSE   EQU    $AE34
                  GDBUFS  EQU    $D539
                  PTRGET  EQU    $DFE3
                  GETSPA  EQU    $E452
                  COUT1   EQU    $FDF0
                  SETKBD  EQU    $FE89
                  SETVID  EQU    $FE93

                          ORG    $300

0300 A960        BEGIN   LDA    #RTS
0302 8D34AE              STA    PAUSE
0305 A900                LDA    #KEYBUF
0307 85FE                STA    COUNT
0309 85FF                STA    FLAG
030B 2089FE              JSR    SETKBD
030E A926                LDA    #STORE
0310 8536                STA    CSW
0312 A903                LDA    /STORE
0314 8537                STA    CSW+1
0316 20EA03              JSR    SETPTR
0319 A900        RESTOR  LDA    #KEYBUF
031B 8D2703              STA    STORE+1
031E A902                LDA    /KEYBUF
0320 8D2803              STA    STORE+2
0323 C6FF        BACK    DEC    FLAG
0325 60                  RTS
0326 8D0002      STORE   STA    KEYBUF
0329 EE2703              INC    STORE+1
032C E6FF                INC    FLAG
032E F00C                BEQ    CONT
0330 20F0FD              JSR    COUT1
0333 2089FE              JSR    SETKBD
0336 2093FE              JSR    SETVID
0339 4CEA03              JMP    SETPTR
033C C98D        CONT    CMP    #RETURN
033E D0E3                BNE    BACK
0340 AE2703              LDX    STORE+1
0343 CA                  DEX
0344 2039D5              JSR    GDBUFS
0347 AD2703              LDA    STORE+1
034A 2052E4              JSR    GETSPA
034D AC2703              LDY    STORE+1
0350 A900                LDA    #KEYBUF
0352 855E                STA    INDEX
0354 A902                LDA    /KEYBUF
0356 855F                STA    INDEX+1
0358 88          LOOP    DEY
0359 B15E                LDA    (INDEX),Y
035B 9171                STA    (FRESPC),Y
035D 98                  TYA
035E D0F8                BNE    LOOP
0360 A5B8                LDA    TXTPTR
0362 48                  PHA
0363 A5B9                LDA    TXTPTR+1
0365 48                  PHA
0366 A98D                LDA    #DATA
0368 85B8                STA    TXTPTR
036A A903                LDA    /DATA
036C 85B9                STA    TXTPTR+1
036E 20E3DF              JSR    PTRGET
0371 68                  PLA
0372 85B9                STA    TXTPTR+1
0374 68                  PLA
0375 85B8                STA    TXTPTR
0377 A000                LDY    #$0
0379 AD2703              LDA    STORE+1
037C 9183                STA    (VARPNT),Y
037E C8                  INY
037F A571                LDA    FRESPC
0381 9183                STA    (VARPNT),Y
0383 C8                  INY
0384 A572                LDA    FRESPC+1
0386 9183                STA    (VARPNT),Y
0388 E6FE                INC    COUNT
038A 4C1903              JMP    RESTOR
038D                     MSB OFF
038D 204141      DATA    DC     C' AA$('
0390 2428
0392 E2                  DC     H'E2'
0393 283235              DC     C'(254)):'
0396 342929
0399 3A
                         END
```

Listing 2

```
10  TEXT : HOME
20  PRINT  CHR$ (4)"BLOAD CAT.OBJ,A$300"
30  GOTO 120
40  INVERSE : PRINT  LEFT$ (Q$,2);: NORMAL
50  PRINT  MID$ (Q$,3);: RETURN
60  IF CARD = 3 THEN  PRINT  CHR$ (12): RETURN
70  HOME : RETURN
80  IF J / (CARD * 2 / 3 + 2) <  >
     INT (J / (CARD * 2 / 3 + 2)) THEN  PRINT
90  RETURN
100 J$ = "": IF J < 10 THEN J$ = "0"
110 J$ = J$ +  STR$ (J): RETURN
120  DIM A$(100),AA$(100),B$(100),I$(100),T$(100),
     S$(100),R$(100): TEXT : HOME
130 CARD = 3: REM  80 COLUMN CARD SLOT:
     SET TO 0 FOR NORMAL 40 COLUMN DISPLAY
140  CALL 768
150  PRINT  CHR$ (4)"CATALOG"
160  PR# CARD: CALL 1002
170 COUNT =  PEEK (254) - 1
180  PRINT : PRINT COUNT" CATALOG ENTRIES ON "AA$(2)
190  FOR J = 4 TO COUNT
200 Z$ =  MID$ (AA$(J),2,1)
210  IF Z$ = "T" THEN T = T + 1: GOSUB 100:T$(T) = J$ +
     MID$ (AA$(J),8,18): GOTO 270
220  IF Z$ = "I" THEN I = I + 1: GOSUB 100:I$(I) = J$ +
     MID$ (AA$(J),8,18): GOTO 270
230  IF Z$ = "A" THEN A = A + 1: GOSUB 100:A$(A) = J$ +
     MID$ (AA$(J),8,18): GOTO 270
240  IF Z$ = "B" THEN B = B + 1: GOSUB 100:B$(B) = J$ +
     MID$ (AA$(J),8,18): GOTO 270
250  IF Z$ = "S" THEN S = S + 1: GOSUB 100:S$(S) = J$ +
     MID$ (AA$(J),8,18): GOTO 270
260  IF Z$ = "R" THEN R = R + 1: GOSUB 100:R$(R) = J$ +
     MID$ (AA$(J),8,19)
270  NEXT
```

(continued)

Listing 2

```
280 SP$ = "          "
290 T$ = "Text Files" +  LEFT$ (SP$,10)
300 I$ = "Integer" +  LEFT$ (SP$,13)
310 A$ = "Applesoft" +  LEFT$ (SP$,11)
320 B$ = "Binary" +  LEFT$ (SP$,14)
330 S$ = "Source" +  LEFT$ (SP$,14)
340 R$ = "Relocatable" +  LEFT$ (SP$,9)
350  GOSUB 60: HTAB CARD * 6 + 9: PRINT "——CATALOG——"
360  IF  NOT T THEN 380
370  PRINT T$;: FOR J = 1 TO T:Q$ = T$(J):
     GOSUB 40: NEXT : GOSUB 80
380  IF  NOT I THEN 400
390  PRINT I$;: FOR J = 1 TO I:Q$ = I$(J):
     GOSUB 40: NEXT : GOSUB 80
400  IF  NOT (A) THEN 420
410  PRINT A$;: FOR J = 1 TO A:Q$ = A$(J):
     GOSUB 40: NEXT : GOSUB 80
420  IF  NOT B THEN 440
430  PRINT B$;: FOR J = 1 TO B:Q$ = B$(J):
     GOSUB 40: NEXT : GOSUB 80
440  IF  NOT S THEN 460
450  PRINT S$;: FOR J = 1 TO S:Q$ = S$(J):
     GOSUB 40: NEXT : GOSUB 80
460  IF  NOT R THEN 480
470  PRINT R$;: FOR J = 1 TO R:Q$ = R$(J): GOSUB 40: NEXT
480  VTAB 23
490  INPUT "WHICH FILE TO RUN? ";A$
500 A =  VAL (A$)
510  IF A = 0 THEN  HOME : END
520  IF  MID$ (AA$(A),2,1) = "S" OR  MID$ (AA$(A),2,1)
     = "R" THEN  PRINT "YOU CAN'T RUN AN 'R' OR 'S'
     TYPE FILE": HOME : END
530  IF  MID$ (AA$(A),2,1) = "B" THEN A$ = "BRUN"
540  IF  MID$ (AA$(A),2,1) = "A" OR  MID$ (AA$(A),2,1)
     = "I" THEN A$ = "RUN"
550  IF  MID$ (AA$(A),2,1) = "T" THEN A$ = "EXEC"
560  PRINT  CHR$ (4);A$; MID$ (AA$(A),8)
570  REM  END
```

MICRO™

"300:A9 60 ... etc.", entering all the hexadecimal code. After a RETURN you need only to type ":" to continue entering data. The monitor keeps track of the addresses. Type "300L" to check for mistakes and save the code with "BSAVE CAT.OBJ,A$300,L$39A".

Type in the BASIC program as listed (listing 2). Line 130 provides for an 80-column card in slot 3. This allows the menu to display the directory entries four across instead of two. If you don't have an 80-column card, set CARD = 0 or delete the line altogether. For debugging, you might want to add a line 145 STOP to the program. If it never gets there, recheck your binary file. Also, I recommend that you run the program for the first time with a disk that you don't particularly care about since it is possible, although unlikely, to crash a disk.

The routine at lines 200-270 checks the file-type and assigns the filename to an array of that file-type. This sorts the directory by file-type so that the listing can be printed in file-type order. If you do not use Source- or Relocatable-type files, there are several lines that can be omitted from the program: 250,260, 330,340,440,450,460,470 (renumber 480 to 440). Lines 350-490 print the directory file names and prompt for a choice. Lines 520-560 check for file-type and send the appropriate command to DOS to execute the file. A change to allow RAM card owners to boot up in Applesoft and load only Integer BASIC when needed would be to add code between 540 and 550 to load Integer on file-type "I".

This routine is very fast; if you name the program "HELLO" so that it will run on boot-up, it is probably faster than booting and typing "CATALOG".

Random Number Generator

by Bill Walker

A theoretical basis for the generation of uniformly distributed pseudo-random numbers.

Random number generators play a large part in many computer applications. These applications can range from making sure not all of your Klingons are consistently destroyed, to controlling computer simulations of real events.

Many of the languages currently available for computers (personal or mainframe) contain some facility to generate random numbers. Most of these generators function by accessing some segment of memory that is likely to change frequently, and using the number thus obtained to generate a number that is claimed to be random. This process is not always completely successful.

The problem is that true randomness is hard to attain. Most physical phenomena are actually controlled by (perhaps unseen) forces that destroy true randomness. About the best that we can hope for is to generate a sequence of "pseudo-random" numbers, that behaves in a manner sufficient for our purposes.

This article will provide insight into random number generators, which are present in many computer languages, and to present some algorithms for generating pseudo-random sequences of numbers in situations where it is necessary for the programmer to "roll their own." The discussion includes term definition, and the presentation of a widely used method for generating pseudo-random sequences. We will write a short program to implement the algorithm presented, and use it to shuffle a card deck.

Definitions

For the purposes of this article, we will regard true random sequences of numbers as unattainable. (Even white noise usually has some organization to it.) We will concentrate instead on the concept of a pseudo-random sequence.

Think of a *pseudo-random sequence* as a bag full of numbers. There are only so many numbers in the bag, but if we reach in the bag and pull out a number, there is no number which is more likely to be selected. That is, each number in the bag has an equal chance of being pulled from the bag.

Each time we select a number from the bag, we will make use of it in whatever application that we will, and then return the number to the bag before we choose another number. It is clear that we will eventually choose the same number twice.

If the bag is a *smart bag*, it will be sure that we use all of the numbers in the bag before it allows any number to be chosen twice. The fact that the smart bag will also present the numbers to us in some predetermined order should not be apparent. The numbers coming from the bag are said to form a pseudo-random sequence since the probability that a particular number is selected is the same as the probability for any other number.

The number of numbers that are selected before the selected numbers start to repeat themselves is called the *period* of the pseudo-random number generator. The period may possibly be as long as the total number of numbers in the bag, but can never exceed however many numbers are in the bag.

To explore the replacement of the smart bag with a computer algorithm, we need to define two more terms. The first term, *prime number*, is an integer (whole or counting number) that cannot be divided evenly by any positive integers other than itself and 1. For instance, the integers 3,5,7,11,13,17, and so on are prime numbers, while 2,4 and 15 are not prime. It is clear that a prime number must be an odd number.

A second related term is that of *relatively prime numbers.* Two positive integers are relatively prime if the smaller will not divide evenly into the larger one. An example would be the integers 2 and 9, neither of which happen to be prime, but these two numbers are relatively prime, since 2 does not divide evenly into 9.

The Algorithm

We will present an algorithm called a *multiplicative linear congruential pseudo-random number generator.*" This particular algorithm is carefully disected and rigorously presented in Knuth's *The Art of Computer Programming*, Volume 2, page 16. If you are a professional programmer, you need access to this fine text.

The heart of the algorithm is the recursive formula:

$$X(n+1) = (A * X(n) + C) \text{ MOD } M$$

where X(0), A, C, and M will be specified. The trick is to specify these quantities correctly. The following theorem appears in Knuth's work.

Theorem

The above formula yields a pseudo-random sequence of length M (i.e. has period M) if and only if

1. C is relatively prime to M
2. B = A − 1 is a multple of P for every prime P which evenly divides M
3. if M is a multiple of 4, then B is also a multiple of 4.

An especially convenient choice of M is to take M as some power of 2. This can make programming in some languages, such as assembly language,

(Continued on page 47)

Tactical Armor Command

One last tug to the helmet strap—a reassuring glance at the line of powerful steel monsters and you know all is ready. From your command hatch you raise your hand and order "forward"! The air suddenly fills with the roar of engines and the rumble of treads as the mighty dreadnaughts of the land start forward. You command a team of your country's finest armored vehicles in a mission to search and destroy the enemy.

T.A.C. is a game of World War II tactical armored combat. **You** pick a nation (from among the four major combatants—Britain, U.S.A., Germany and Russia). **You** build a combat team from their most powerful tanks, assault guns and tank destroyers. **You** command the team you've created in major operations against like forces of the enemy.

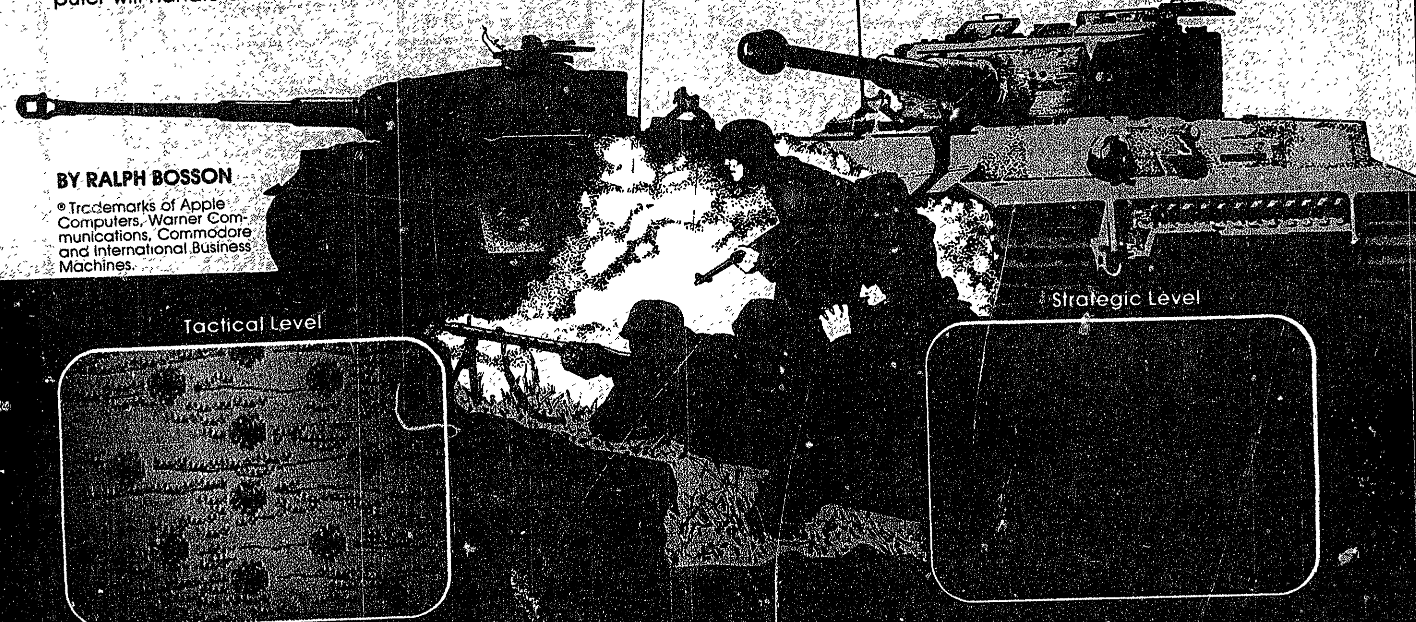
All the famous vehicles of the second world war are here—Tigers, Panthers, Shermans and JS II's; Jagdpanthers, SU 152's, Fireflies and T 34's, just to name a few. They have all been thoroughly researched and their important features programmed into the game. Each vehicle is distinguished by such elements as armor thickness (rear and flanks as well as front), fire power, speed, acceleration and gun traverse. Even minor points like fuel tank location can be critical.

The computer handles all the technical details. This lets you concentrate on making the same kinds of decisions the real life tank commanders made. You search for the enemy, set your speed, aim your gun and knock out the enemy. The computer will handle all the rest.

Here are just some of the exciting features:

- **T.A.C.** can be played solitaire against the computer or as a two player (or two team) game with the computer as mediator.
- The most important armored vehicles of Britain, Russia, U.S. and Germany are available to command—40 in all.
- Choose from five different scenarios to play. Actions range from open meeting engagements to assaults against prepared positions.
- You pick the sides. You choose the weapons. A simple purchasing system has been provided to let you "buy" what you want in balance with your opponent.
- The results of combat are determined by the computer. It factors such critical elements as range, armor thickness (front, rear and flanks), tracking time, the speed and maneuvers of both the firing and target units, visibility and weapon adjustment to determine weapon accuracy.
- Special options include hidden movement, improved positions, smoke mortars, minefields, close assaults, overruns and indirect fire.

T.A.C. on diskette retails for $40.00 and can be played on the following computers: Apple® II's with 48K (Mockingboard™ Sound Enhanced!). Atari's® with 48K. Commodore® & IBM® versions coming this fall.



BY RALPH BOSSON

® Trademarks of Apple Computers, Warner Communications, Commodore and International Business Machines.

Available at finer computer stores everywhere.
Or call TOLL FREE: 1 (800) 638-9292 for fast credit card purchase. Price: $40. Ask for Operator M.

AH QUALITY microcomputer games®
A DIVISION OF
The Avalon Hill Game Company
4517 Harford Road • Baltimore, MD 21214

(Continued from page 44)

very easy. Suppose we took M = 256 (2 to the 8th). Then 2 is the only prime P that divides M, so we can take B to be any even number, and therefore A must be odd. Since we must have B, a multiple of 4, let us choose, for instance, B = 12 or A = 13. In this case, C can be any number other than 2. For convenience we will take C to be 3.

We now have the formula:

```
X(n + 1) = (13*X(n) + 3) MOD 256
```

The following code would generate an array containing 256 pseudo-random numbers, ranging from 0 to 255, and would contain no repeated numbers.

```
X(0) := SEED;
FOR I := 1 TO 255 DO
     X(I) := (13*X(I-1) + 3) MOD 256;
```

The SEED is any integer between 0 and 255. Many small computers arrange SEED to be the value of some byte that often changes, such as the last byte present on an I/O port. If the same number is used for SEED in a second run of the program segment, the resulting sequence of numbers will be exactly the same as before, but if a different SEED is used, the resulting sequence will be different. We see that since there are 256 possible choices for the SEED, the algorithm can generate 256 different pseudo-random number sequences.

If we were to choose M to be much larger, say M = 1024 (2 to the 10th), we could generate 1024 different pseudo-random sequences with the same algorithm.

An Example

It is clearly to our advantage to choose M above as some power of 2. However, to illustrate that this is not necessary, we will write an example card deck shuffling program.

We will number the cards in a standard card deck from 1 to 52. For instance, we might have

```
DECK[1] := 'ACE OF SPACES'
DECK[2] := 'ACE OF CLUBS'
```

and so on. To shuffle the deck we will just present the cards by choosing the subscripts of the DECK using a pseudo-random number generator of our own design. Clearly, we must have a period of 52, since we can't afford to deal the same card twice (!).

We choose M = 52, to get the period that we desire. We are now required to choose C so that C and M are relatively prime. Since M is an even number, it suffices to choose C as any odd number. We will pick C = 3 since that is as good as any.

The choice of the mutiplier A is not as obvious. According to the theorem A − 1 must be a multiple of every prime that divides M. We note that M is 2 * 2 * 13, so that A − 1 must be a multiple of 2 and of 13. We also note that 4 is a divisor of M, so that A − 1 must also be a multiple of 4. All of this means that A − 1 must be 4 * 13, or 52. So we take A to be 53.

We now have the following algorithm:

```
SUB := SEED;
FOR I := 1 TO 52 DO
     BEGIN
     WRITELN (DECK[SUB + 1]);
     SUB := (53*SUB +3) MOD 52;
     END
```

This will deal 52 cards in a random order. There are 52 possible such dealings. We can choose one by specifying the value of SEED, which can be the value of some memory location that varies radically in time, or some other number essentially independent of time.

The subscript of "SUB + 1" was necessary, since the numbers generated vary between 0 and 51, and not between 1 and 52 as might be supposed at first glance. (It is clear that 52 MOD 52 is 0).

Testing the Generator

Assuring ourselves that the pseudo-random number generators that we invent are as "random" as possible is not an easy task. Poor choices of A, C, and M can lead to very poor generators. One famous example occurs in an old friend RANDU from unit-record processing days. On many machines, this generator used one of the poorest possible choices of A, and thus provided bias to many unsuspecting users.

If you run the card shuffler program segment above, you will discover that the generator tends to produce "runs" or sequences of numbers that do not appear to be random in nature. The card shuffler will not pass a "visual inspection" for randomness. How do you fix it? Just "fiddle" with the multiplier A and the SEED. You can improve considerably on the generator presented above by trial and error methods. Of course, trial and error is not adequate for many purposes, and we would desire that there exist a more rigorous method of assuring randomness. Again, we find Professor Knuth coming to the rescue.

Knuth devotes about 70 pages to the discussion of testing pseudo-random number generators such as the one above. Testing algorithms are presented and discussed in mathematical environment. If you are using random-number generators, either of your own design or as a built-in feature of language, you should acquire and read Knuth's second volume. If you are only zapping Klingons, it is not really necessary of course, but if you are building bridges for the public to walk on, you probably should make a careful study, if only for your own benefit.

It is interesting to note that RANDU, the built-in random number generator that was the standard in the industry for many years, flunks some of Knuth's tests miserably.

The random number generators present in some languages may be inadequate for critical tasks, or may be absent altogether. This article has presented an algorithm that allows the user to develop pseudo-random number generators that are adequate for most hobbyist purposes. The tests, which can be found in the literature, are capable of assuring that pseudo-random number generators are adequate for a given purpose.

The following rules make design of a pseudo-random number generator easy.

1. Choose a period equal to a power of 2.
2. Choose A to be equal to a power of 2 plus 1.
3. Choose C to be odd.

Suggested Reading

1. *The Art of Computer Programming*, Volumes i,ii,iii, by Donald Knuth.
2. *Algorithms + Data Structures = Programs*, by N. Wirth.
3. *A Structured Approach to Pascal*, by Bill Walker.
4. "Discrete Event Simulation," by Bill and Anita Walker (MICRO 56:21)

Bill Walker is Assistant Professor of Electrical Engineering and Computer Science at the University of Oklahoma. You may contact him at Box 2806, Norman, OK 73070.

MICRO

SEARCHING REVEALED: LINEAR SEARCH

by Richard C. Vile, Jr.

Linear Search requires:
Apple II
Listings 1, 2, 4 and 6 require Applesoft.
Listings 3 and 5 require Pascal.

Searching is a technique used in many computer programs. More often than not, a list of some sort is searched. The list may consist of a variety of different kinds of information. For example:

- A mailing-list program may use a list of names, addresses, or both.
- In a program to maintain bowling scores, it may be necessary to search a list of names for the name of a specific player.
- Computer language translators usually maintain a list of identifiers or keywords that must frequently be searched to distinguish names reserved by the language from those created by the programmer.
- In an interactive game, a list of command words may need to be searched in order to match the commands typed by the player.

In all these examples, the lists that are searched consist of what are usually called *character strings*, or simply *strings*. Strings are represented in various ways depending on the programming language you use:

APPLESOFT — String variables hold one string apiece: A$.
Pascal — String variables hold one string apiece: S = 'Hi'.
Assembly language — Strings are just sequences of bytes in memory.
Integer BASIC — String variables are arrays of characters and usually hold one string apiece.

Of course, in order to represent a list of items in a program, you must be able to store many strings at once. This is done using an array of strings. Not all Apple languages directly allow for arrays of strings — APPLESOFT and Pascal do; Integer BASIC and Assembly language do not. For this reason, my examples will center on the former two languages.

Linear Search in General

The general technique of linear search assumes the existence of a linearly ordered collection of items. In these examples, I consider arrays of strings. Any collection of "things" organized in a lineup of some sort may be subjected to linear search. Some examples from real life are a pile of magazines on a coffee table, a shelf of books, a bin of records on sale at a discount store, your mailbox full of letters and junk mail, a poorly organized collection of recipes on 3×5 cards, and the want ads in your local newspaper.

In all of these examples, if you were searching for a specific item such as last month's MICRO, an old Beatle's album, your income tax refund check, a recipe for Quiche Lorraine, or a For Sale ad for a used computer, you might be apt to start at the top and search through the collection one item at a time. You would continue until you found what you were looking for or until you ran out of items.

In some cases you might take advantage of extraneous information to speed up your search. For instance you might remember the color of the cover of last month's MICRO. You could then limit your search to magazines whose covers were of that color. You might look through your mail for an official-looking government envelope; or you might look for the word "COMPUTER" in capital letters in the want ads.

Because you are human, you have sophisticated pattern-matching abilities with which a computer cannot yet compete. A computer, searching a list of items, is not able to use such cues in most cases. It has to take each item in turn to see whether or not it is the one being sought. This is always true if the list being searched has no other structure than that of a list. In future articles I will discuss to what extent a computer might take advantage of "extra structure." For now, however, I make no assumptions. The computer lists are simply big unordered piles — like a collection of twenty years' worth of *Life* magazines well shuffled from use.

Here's how to search:

1. First we ask, "Are there any more things left in the pile for which we are searching?" If yes, continue the search; i.e., do step 2. If no, stop the search.
2. Is the next thing in the pile the item for which we are looking? If yes, we succeeded, so quit. If no, do step 3.
3. Put aside the item at which we just looked and rejected. Continue the search from step 1.

The following short names can be given to the three steps in the above procedure: 1. TEST, 2. COMPARE, and 3. LOOP. I shall refer to these "ingredients" in my discussion below.

Linear Search in Applesoft

Listing 1 shows a simple Applesoft program illustrating linear search. The search itself is done in the subroutine beginning at line 1000. The rest of the program makes sure there is a list to be searched and there are items to be searched. The subroutine in listing 1 is written in a primitive style in order to illustrate the components of the linear search explicitly:

```
TEST: IF J > 100 THEN RETURN
COMPARE: IF (ST$(J) = P$) THEN
            F0 = 1
LOOP: J = J + 1 : GOTO 1010
```

Listing 2 uses the BASIC FOR statement as an alternate method.

Linear Search in Pascal

Listing 3 shows another simple program, this time in Apple Pascal. The program is similar in spirit to that of listing 1, and the list to be searched contains the names of the U.S. Presidents.

Study listing 3 and then compare it to listing 1. You should be able to see many interesting differences in the programming style of the two languages, Applesoft and Apple Pascal. If you are just beginning to program in Pascal, this suggests one excellent way to learn the language; *try* to rewrite some of your BASIC programs using Pascal. Try is the key word here as there are some programs that rely too heavily on the memory layout of the Apple to be translated. See how close you can come. And concentrate on translating the spirit of the program — i.e., what it *does.* Don't try to make a literal line-for-line translation or you'll just get bogged down and probably give up.

Sentinels — More Efficient Linear Search

The test for completion of a linear search, "Are we out of items to consider?", is not conceptually part of the search itself. It seems like unwanted extra baggage. It really is as you soon will see.

Suppose you knew ahead of time that what you were searching for definitely was one of the items in the collection being searched. Or, to put it another way, suppose you knew at the start that a successful search was guaranteed. Then the TEST part of the procedure would be superfluous. You might think the whole search would be superfluous! Leaving that issue aside for a moment, let's see if we can think of a way to guarantee that all your linear searches have happy endings.

Figure 1 gives the basic idea — an extra location in the search collection. Why would you want to increase the number of items to be searched? To guarantee success, of course. You will use the extra location to store a copy of the item for which you are looking. Then if that item turns out not to be in the collection *proper*, you can still find it in the extra location at the end. Therefore you won't have to worry about TESTing whether or not any items are left. At the worst, you will find what you are looking for just before you run out of items to consider.

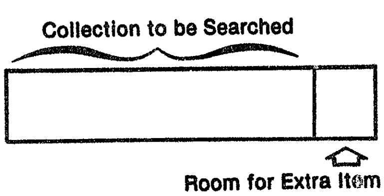


***Figure 1:* Sentinel Location**

The extra item added to the collection is known as a *sentinel* since it stands guard against the possibility of failure.

You now have a slightly different problem to solve because there are now two possible ways to succeed.

1. Find the sentinel.
2. Find the item for which you are looking before you get to the sentinel.

In the first case, even though you succeed in one sense, you fail in the larger sense. Case 2 could be dubbed a real success. After you succeed (which you know you will since you have a sentinel), you check to see whether or not you are at the sentinel location. If not, then you *really* succeed. If so, then you were only helped over the finish line. Real success awaits in some future search.

Sentinel Searching in Applesoft and Pascal

Listings 4 and 5 show the linear search subroutines of listings 1 and 3 augmented by the use of a sentinel location. Notice that in each case the array used to hold the collection to be searched must be given an extra location. The first step in the search procedure is then to store the item being searched in the extra location (at the end of the regular array).

Screen Searching

Listing 6 presents the linear search without a sentinel. It is programmed in Applesoft and runs on the screen before your very eyes. It is almost entirely self-explanatory, but if at any time you think it is stuck, just hit RETURN and it is likely to continue on its merry way.

EXERCISE: Modify the program of listing 6 to use a sentinel location.

Coming Up

In the next article in this series, I will discuss the mechanism of the Binary Search and present an on-screen demonstration similar to the Linear Search demo of listing 6.

You may contact Dr. Vile at
3467 Yellowstone Dr., Ann Arbor,
MI 48105.

(Listings begin on page 50)

Listing 1

```
5  DIM ST$(50),OK(50)
10  FOR I = 1 TO 50: READ ST$(I): NEXT I
15  FOR I = 1 TO 50:OK(I) = Ø: NEXT I
20 C = Ø
100  REM ================
101  REM = MAIN PROGRAM =
102  REM ================
105  GOSUB 200
106  INPUT "? ";P$
107  IF P$ = "BYE" THEN 180
110 FO = Ø: GOSUB 1000
115  IF FO = 1 THEN 150
120  PRINT P$;" IS NOT A STATE"
125  PRINT : GOTO 106
150  IF OK(J) = Ø THEN OK(J) = 1:C = C + 1:
      PRINT "GOOD!": GOTO 105
155  PRINT "YOU ALREADY NAMED THAT ONE!"
160  PRINT "YOU NAMED ";C;" STATES WITHOUT REPEATING"
165  PRINT "GO AGAIN?"
170  INPUT A$: IF A$ = "Y" OR A$ = "YES" THEN 15
180  PRINT "YOU GOT ";C;" STATES"
185  END
200  REM ================
201  REM = INSTRUCTIONS =
202  REM ================
205  HOME : VTAB 5: PRINT "WELCOME TO THE GAME OF THE STATES": PRINT
210  PRINT "GUESS AS MANY STATES AS YOU CAN"
211  PRINT "WITHOUT REPEATING YOURSELF."
212  PRINT "GOOD LUCK..."
215  VTAB 23: HTAB 5: PRINT "TO START, PRESS RETURN";
220  GET A$
225  HOME : RETURN
1000  REM ============================
1001  REM = LINEAR SEARCH SUBROUTINE =
1002  REM ============================
1005 J = 1
1010  IF J > 50 THEN  RETURN
1015  IF (ST$(J) = P$) THEN FO = 1: RETURN
1020 J = J + 1
1025  GOTO 1010
2000  REM ==================
2001  REM = LIST OF STATES =
2002  REM ==================
2005  DATA  MAINE, VERMONT, NEW HAMPSHIRE
2006  DATA   MASSACHUSSETTS, CONNECTICUT, RHODE ISLAND
2007  DATA  NEW YORK, PENNSYLVANIA, DELAWARE
2008  DATA  MARYLAND, VIRGINIA, NORTH CAROLINA
2009  DATA  SOUTH CAROLINA, GEORGIA, FLORIDA
2010  DATA  OHIO, WEST VIRGINIA, KENTUCKY
2011  DATA  TENNESSEE, ALABAMA, MISSISSIPPI
2012  DATA  MICHIGAN, INDIANA, ILLINOIS
2013  DATA  WISCONSIN, LOUISIANA, ARKANSAS
2014  DATA  MISSOURI, IOWA, MINNESOTA
2015  DATA  WASHINGTON, CALIFORNIA, OREGON
2016  DATA  IDAHO, NEVADA, ARIZONA
2017  DATA  NEW MEXICO, UTAH, MONTANA
2018  DATA  WYOMING, COLORADO, TEXAS
2019  DATA  OKLAHOMA, NEBRASKA, NORTH DAKOTA
2020  DATA  SOUTH DAKOTA, ALASKA, HAWAII
2021  DATA  KANSAS, NEW JERSEY
```

Listing 2

```
1000  REM ============================
1001  REM = LINEAR SEARCH SUBROUTINE =
1002  REM ============================
1010  FOR J = 1 TO 50
1015  IF (ST$(J) = P$) THEN FO = 1: RETURN
1020  NEXT J
1025  GOTO 1015
1049  RETURN
```

Listing 3

```
PROGRAM USPresidents;

TYPE
     presname = STRING[25];

VAR
     presidents:     ARRAY[1..40] OF presname;
     prex:           presname;
PROCEDURE init1prexies;
BEGIN
     presidents[1]     := 'George Washington';
     presidents[2]     := 'John Adams';
     presidents[3]     := 'Thomas Jefferson';
     presidents[4]     := 'James Madison';
     presidents[5]     := 'James Monroe';
     presidents[6]     := 'John Quincy Adams';
     presidents[7]     := 'Andrew Jackson';
     presidents[8]     := 'Martin Van Buren';
     presidents[9]     := 'William Henry Harrison';
     presidents[10]    := 'John Tyler';
     presidents[11]    := 'James Polk';
     presidents[12]    := 'Zachary Taylor';
     presidents[13]    := 'Millard Fillmore';
     presidents[14]    := 'Franklin Pierce';
     presidents[15]    := 'James Buchanan';
     presidents[16]    := 'Abraham Lincoln';
     presidents[17]    := 'Andrew Johnson';
     presidents[18]    := 'Ulysses S. Grant';
     presidents[19]    := 'Rutherford B. Hayes';
     presidents[20]    := 'James Garfield';
END  { PROCEDURE init1prexies };

PROCEDURE init2prexies;
BEGIN
     presidents[21]    := 'Chester A. Arthur';
     presidents[22]    := 'Grover Cleveland';
     presidents[23]    := 'Benjamin Harrison';
     presidents[24]    := 'Grover Cleveland';
     presidents[25]    := 'William McKinley';
     presidents[26]    := 'Theodore Roosevelt';
     presidents[27]    := 'William Howard Taft';
     presidents[28]    := 'Woodrow Wilson';
     presidents[29]    := 'Warren G. Harding';
     presidents[30]    := 'Calvin Coolidge';
     presidents[31]    := 'Herbert Hoover';
     presidents[32]    := 'Franklin Delano Roosevelt';
     presidents[33]    := 'Harry S. Truman';
     presidents[34]    := 'Dwight D. Eisenhower';
     presidents[35]    := 'John F. Kennedy';
     presidents[36]    := 'Lyndon B. Johnson';
     presidents[37]    := 'Richard M. Nixon';
     presidents[38]    := 'Gerald R. Ford';
     presidents[39]    := 'Jimmy Carter';
     presidents[40]    := 'Ronald Reagan';
END { PROCEDURE init2prexies };

FUNCTION search (VAR s : presname) : BOOLEAN;
VAR
   i:        INTEGER;
   found:    BOOLEAN;
BEGIN
   found          := FALSE;
   i              := 1;

   WHILE (( i <= 40) AND (NOT found))
   DO
   BEGIN
        IF s = presidents[i]
        THEN
             found := TRUE;
        i := i + 1;
   END;

   search := found;
END;

BEGIN
     init1prexies;
     init2prexies;

     WHILE prex <> 'quit'
     DO
     BEGIN
          WRITELN ('Name a President');
          READLN  (prex);
          IF search  (prex)
          THEN
          BEGIN
               WRITELN ('Good! You got one!');
               WRITELN ('Try again?');
               READLN  (prex);
          END;
     END { WHILE prex <> 'quit' };
END { PROGRAM USPresidents }.
```

(Continued on page 52)

Listing 4

```
1000  REM ============================
1001  REM = LINEAR SEARCH SUBROUTINE =
1002  REM ============================
1005 ST$(51) = P$
1010 J = 1
1015  IF (ST$(J) = P$) THEN 1030
1020 J = J + 1
1025  GOTO 1015
1030  IF J < 51 THEN FO = 1
1049  RETURN
```

Listing 5

```
FUNCTION search (VAR s : presname) : BOOLEAN;
VAR
    i:          INTEGER;
    found:      BOOLEAN;
BEGIN
    found            := FALSE;
    presidents[41] := s;
    i                := 1;

    WHILE (s <> presidents[i])
    DO
    BEGIN
        i := i + 1;
    END;

    search := (i <> 41);
END;
```

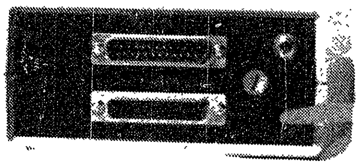
Listing 6

```
5  DIM IT$(30),SE$(16),PI(30)
50 DV = 75:TRY = Ø
100  REM ================
101  REM = MAIN PROGRAM =
102  REM ================
105  GOSUB 200: REM  INSTRUCTIONS
110  GOSUB 300: REM  SETUP
115  GOSUB 400: REM  DISPLAY
117  IF A$ <>  CHR$ (13) THEN TRY = TRY + 1: GOTO 110
120  FOR L = 1 TO 16
123  GOSUB 700: REM  REMOVE OLD TRY
125 FO = Ø: GOSUB 500: REM  TRY
130  IF FO = 1 THEN 150
135  VTAB 22: HTAB 30: FLASH : PRINT "NO";: NORMAL
138  FOR DE = 1 TO 1000 - 75 * L - 100 * TRY:
      NEXT DE: GOSUB 800
140  NEXT L
141  GOSUB 700: GOSUB 800
142  VTAB 24: HTAB 1: CALL  - 868
143  VTAB 22: CALL  - 868: INVERSE
144  HTAB 5: PRINT S$;" IS NOT ON THE LIST";:
      NORMAL : GOSUB 900 REM  BUZZ
145  GOSUB 800: GOSUB 800: GOTO 165
150  VTAB 22: HTAB 30
155  CALL  - 868: FLASH : PRINT "YES";: NORMAL
156  GOSUB 800: GOSUB 800
160  VTAB 24: HTAB 1: CALL  - 868
161  VTAB 22: CALL  - 868: INVERSE
162  HTAB 5: PRINT S$;" WAS FOUND AT ";L;
163  NORMAL : GOSUB 800: GOSUB 800
165  GOSUB 600
170 TRY = TRY + 1
175  IF A$ =  CHR$ (13) THEN 110
199  END
200  REM ================
201  REM = INSTRUCTIONS =
202  REM ================
205  HOME : SPEED=  25
210  PRINT "------ LINEAR SEARCH DEMO ------": PRINT
215  PRINT "THIS PROGRAM WILL DEMONSTRATE THE"
220  PRINT "TECHNIQUE OF LINEAR SEARCH WITHOUT"
225  PRINT "A SENTINEL LOCATION.  A LIST OF"
230  PRINT "STRINGS WILL BE GENERATED ON THE "
235  PRINT "LEFT SIDE OF THE SCREEN AND A STRING"
240  PRINT "TO BE SEARCHED FOR WILL BE PRINTED"
245  PRINT "AT THE TOP OF THE SCREEN.  THE PROGRAM"
250  PRINT "WILL GO DOWN THE LIST 'LOOKING' FOR"
255  PRINT "A MATCH.  AFTER EACH STEP, IT WILL "
260  PRINT "PAUSE AND POSSIBLY DISPLAY COMMENTS"
265  PRINT "REGARDING ITS PROGRESS.  TO MAKE IT"
270  PRINT "CONTINUE, SIMPLY PRESS THE RETURN KEY."
280  GOSUB 600
299  SPEED= 255: RETURN
300  REM ==============
301  REM = INITIALIZE =
302  REM ==============
305 X =  PEEK (78) + 256 *  PEEK (79)
310  RESTORE : FOR I = 1 TO 30: READ IT$(I): NEXT I
315  FOR I = 1 TO 30:PI(I) = 0: NEXT I
320  FOR J = 1 TO 16
322 I =  INT ( RND (X) * 30 + 1)
324  IF PI(I) THEN 322
326 SE$(J) = IT$(I):PI(I) = 1
328  NEXT J
330 I =  INT ( RND (X) * 30 + 1)
334 S$ = IT$(I)
349  RETURN
400  REM ===========
401  REM = DISPLAY =
402  REM ===========
404  HOME
405  FOR K = 1 TO 16
410  VTAB 3 + K: HTAB 1
412  IF K < 10 THEN  PRINT " ";
414  PRINT K;"  ";SE$(K)
420  NEXT K
450  VTAB 2: HTAB 5
452  CALL  - 868
455  INVERSE : PRINT "LOOKING FOR ===>";
460  FLASH : PRINT "  ";S$;"  "
465  NORMAL : GOSUB 600
470  VTAB 2: HTAB 21: PRINT "  ";S$"  "
499  RETURN
500  REM ===================
501  REM = TRY A MATCH AND =
502  REM = RETURN RESULT.  =
503  REM ===================
505  VTAB 24: HTAB 1: CALL  - 868
508  VTAB 3 + L: HTAB 21: PRINT " = ";S$;"?";
510  VTAB 22: HTAB 5
512  CALL  - 868: GOSUB 850
513  SPEED= 150
514  PRINT "IS IT ";SE$(L);" ?": SPEED= 255
515  FOR DE = 1 TO 1000 - 100 * L - 100 * TRY:
      NEXT DE: GOSUB 850
520  IF SE$(L) <> S$ THEN  RETURN
525 FO = 1
549  RETURN
600  REM =========
601  REM = PAUSE =
602  REM =========
605  VTAB 24: HTAB 1
610  CALL  - 868
620  SPEED= 100
640  PRINT "------";
645  PRINT " PRESS RETURN TO CONTINUE";
650  SPEED= 255
655  GET A$
699  RETURN
700  REM =====================
701  REM = ERASE OLD ATTEMPT =
702  REM =====================
705  IF L > 1 THEN  VTAB 2 + L: HTAB 21: CALL  - 868
749  RETURN
800  REM ==============
801  REM = DELAY LOOP =
802  REM ==============
805  FOR DE = 1 TO 500: NEXT DE
809  RETURN
850  REM ================
851  REM = FASTER DELAY =
852  REM ================
860  FOR DE = 1 TO 200: NEXT DE
899  RETURN
900  REM ========================
901  REM = SOUND FAILURE BUZZER =
902  REM ========================
910 XX =  PEEK ( - 16336) +  PEEK ( - 16336) +
     PEEK ( - 16336) +  PEEK ( - 16336)
915  FOR I = 1 TO 2: NEXT I
920  IF J > 25 THEN  RETURN
925 J = J + 1: GOTO 910
9000  DATA  APPLE II, PET, COMMODORE 64
9001  DATA  APPLE III, VIC 20, TRS-80
9002  DATA    COLOR COMPUTER,RADIO SHACK 16, OSBORNE I
9003  DATA  ALTAIR 8800, CROMEMCO Z2, ATARI 400
9004  DATA  ATARI 800, SUPER PET, IBM PC
9005  DATA   SINCLAIR ZX81, INTERACT, OSI CHALLENGER
9006  DATA  EXIDY SORCERER, TI COMPUTER, DEC RAINBOW
9007  DATA  ZENITH Z100, SGL COMPUTER, IMSAI 8080
9008  DATA  DIGITAL GROUP, ALTAIR 680, ECD MICRO MIND
9009  DATA  HEATH H89, COMPUCOLOR II, HEATH H11
9010  DATA  EPSON HX-20, PENTEL PENPUTER,VICTOR 9000
```

ZANIM SYSTEMS

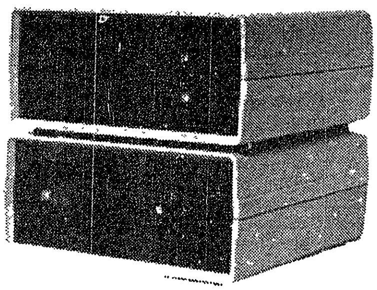
P.O. Box 4364
Flint, Michigan 48504
(313) 233-5731
(313) 233-3125

HOME CONTROL AT YOUR FINGERTIPS



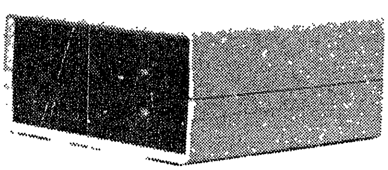
The ZCM-1 is the Master Control module that provides the interface between your computer and our line of Zanim Application Modules. Up to 15 Application Modules can be piggy-backed to the ZCM-1 Master Control module. The ZCM-1 is compatable with any standard RS-232 (serial) interface. A special Master 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.



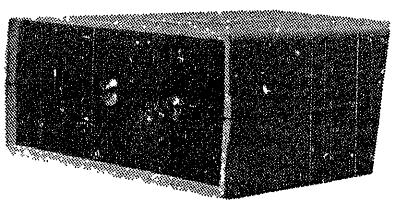
ZAM-1 is the home control interface module that provides a fully versatile computer controlled environment. ZAM-1 can control up to 256 different lamps and appliances in your home or business providing you with an effective and easy to implement energy management and electric control system. No special 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.

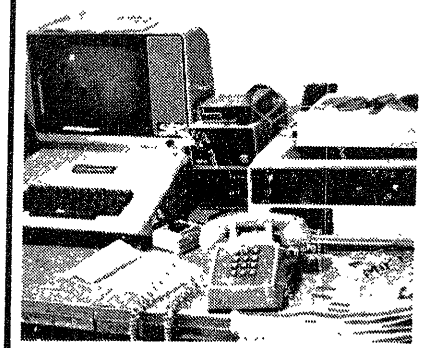


The ZAM-3 is a complete telephone answering and dialing system. It is capable of taking the phone off-hook and dialing a number under computer control or of answering the phone when it rings. With the ZAM-1 Home Control module and the ZAM-2 Security module, the ZAM-3 Phone Dialer module can be integrated 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.

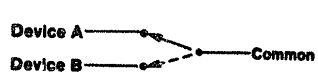
*The ZCM-1/ZCM-1V Master Control module is required to use the ZAM-3/ZAM-3P Phone Dialer module.

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?

____

APPLE 11 IS TRADEMARK OF APPLE COMPUTERS, INC.
CP/M IS REGISTERED TRADEMARK OF DIGITAL RESEARCH, INC.

TRS – 80 IS REGISTERED TRADEMARK OF TANDY CORP.
VIC - 20 AND COMMODORE - 64 ARE REGISTERED TRADEMARKS OF COMMODORE, INC.

Faster Math Operations in Microsoft BASIC

by Peter Hiscocks

Array operations are slow in Microsoft BASIC because they have to be explicitly programmed. For example, to add two 32 element arrays, we require

```
FOR I=0 TO 32
RL(I)=RL(I)+DV(I)
NEXT I
```

which takes about 0.[illegible] seconds.

For each trip around the loop, BASIC has to set a value for I and then find the variables RL(I) and DV(I). We know, however, that the variables follow in sequence, and this can be used to speed things up considerably.

The variables of a floating point array are stored in five-byte lumps, one after another in memory. To operate on two arrays, we set up pointers to the zeroth elements, perform the arithmetic, then increment the pointers by five bytes, continuing until the two arrays are done. Using the ROM routines, the routine given above may be made to execute in 0.066 seconds. As a rough rule of thumb, expect a factor of 6 to 9 increase in speed.

To use this technique we have to be able to find the arrays in memory and use the BASIC floating point arithmetic routines.

The method shown here is accompanied by ROM locations for Commodore PET BASIC version 4, but the ideas should be transferrable to other versions of Microsoft BASIC.

Finding the arrays

As shown in figure 1, arrays grow upward in memory, as they are dimensioned, above simple variables. The creation of a variable or editing of BASIC text will change the shaded areas, so we cannot count on the arrays being at any fixed location. However, the Microsoft routine that finds a variable can be used to do this. Handed the zeroth element as a variable — DV(0), for example — the routine will give us back the memory location of the start of the array.

MADD
requires:
PET/CBM or Commodore 64

Array Storage Format

An array consists of a header, which contains descriptive information about the array, and the body, which contains the array elements. A typical floatng point array header is shown in figure 2. Each block is one byte.

In a multi-dimensional array, the last dimension changes least frequently. For example, DV(32,5) would be stored as DV(0,0) to DV(32,0) followed by DV(0,1) to DV(32,1), and so forth to DV(32,5).

Arithmetic Operations

The arithmetic operations could be programmed in machine language. This would be worthwhile for operations on integer arrays since Microsoft BASIC performs integer arithmetic by converting the number to a floating point, using the floating point routine, and converting back to integer.

For floating point math, the easy way out is to use the Microsoft ROM routines (table 1). These have been well described elsewhere, so this will be a summary.

Mathematical operations take place in one of two, six-byte "accumulators" on page zero. The FAC is located at $5E to $63, and the ARG is located at $66 to

Figure 1: **Array Storage In Ram**

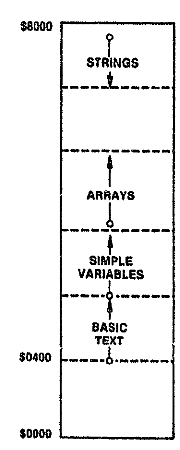


Table 1: Locations and Routines used in MADD and others of interest.

	BASIC 2	BASIC 4	64
Check for a comma: CHKCOM	CDF8	BEF5	AEFD
Evaluate an expression: FRMEVL	CC9F	BD98	AD9E
Move variable to FAC: MOVFM Variable address low byte to AC Variable address high byte to Y	DAAE DAAE	CCD8 CCD8	BBA2 BBA2
Move variable to ARG: CONUPK Variable address low byte to AC Variable address high byte to Y	D998	CBC2	BA8C
Store FAC in memory: MOVMF Variable address low byte to X Variable address high byte to Y	DAE0	CD0A	BBD4
Transfer FAC to ARG	DB18	CD42	BC0C
Transfer ARG to FAC	DB08	CD32	BBFC
Convert FAC to 16 bit integer: AYINT Result in $61 and $62	D09A	C2EA	B1BF
Add FAC = ARG + FAC	D77B	C9A5	B86F
Subtract FAC = ARG − FAC	D736	C989	B853
Multiply FAC = ARG * FAC	D93C	CB66	BA2B
Divide FAC = ARG / FAC	DA20	CC4A	BB12
FACHI	61	61	64
FACLO	62	62	65
VARPNT	44	44	47
OVRFLO	65	65	68
UNDFLO	6D	6D	70

Listing 1

```
                ;***********************************
                ;*
                ;*      madd
                ;*
                ;*      matrix addition program
                ;*
                ;*      by peter hiscocks
                ;*      adds dv(i) to rl(i) with result in rl(i)
                ;*     also stores integer of rl(i) and
                ;*     stores in output table, outble
                ;*     calling syntax- sys gtrl,rl(0),dv(0)
                ;*     maximum 255 elements
                ;*
                ;*     assembled for commodore 64
                ;*     see table for basic 2,4, & vic
                ;*
                ;***********************************
6006            chkcom  =     $aefd   ; check for comma, else syntax error
6006            frmevl  =     $ad9e   ; evaluate expression
                 ; in this case, find
                 ; pointer address to
                 ; start of matrix, in
                 ; varpnt & varpnt+1
6006            movfm   =     $bba2   ; memory (rl) to fac #1
6006            conupk  =     $ba8c   ; memory (dv) to fac #2 (arg)
6006            fadd    =     $b86f   ; sum of fac & arg to fac
6006            movmf   =     $bbd4   ; fac to memory (rl)
6006            ayint   =     $b1bf   ; fac to integer,
                 ;  result in faclo
6006            fachi   =     $64
6006            faclo   =     $65     ; mantissa lsb of fac
6006            varpnt  =     $47     ; pointer to variable
6006            ovrflo  =     $68     ; overflow error flag
6006            undflo  =     $70     ; underflow err flag
                 ;
6000                    *=    $6000
                 ;
6000            outble  =     *+200   ;integer output table
                 ;   64 locations
                 ;
6001            rllo    *=    *+1     ; loop address pointers
6002            rlhi    *=    *+1
                 ;
6003            dvlo    *=    *+1
6004            dvhi    *=    *+1
                 ;
6005            elemen  *=    *+1     ; number of elements in each array
                 ;
```

(continued)

$6A. ROM routines are available to move a variable from its storage location into accumulator (in the process "unpacking" changes the format from five to six bytes). They are available to perform the arithmetic operation, and to repack the variable back into its storage location.

The example of listing 1, MADD, is a Matrix ADD for 32 elements. It stores each result back in memory and also converts it to fixed point format, sending that to an output table OUTBLE for use by a piece of hardware. It's a pretty specialized piece of software, but it shows how all this might go together.

The routine is called from BASIC with an instruction like

SYS XXX,RL(0),DV(0)

where XXX is the start of the machine-language routine in memory. Notice that the same routine XXX may be called with different parameters, or different routines may be created to perform different array operations.

When the SYS XXX instruction is executed by the BASIC interpreter, the BASIC line scanner is left pointing at the first comma.

The routine CHKCOM looks for this and prints "SYNTAX ERROR" if it's not present. The scanner is now pointing at RL(0). Calling the routine FRMEVL (evaluate an expression) will put the location of RL(0) in the page zero locations VARPNT. (It also leaves the value of RL(0) in the FAC, but that's not used here). We can snag the starting location of the array RL(I) by reading VARPNT. The routines CHKCOM and VARPNT may be used again to find the starting location of DV(I).

Loop control for the number of elements in the arrays is built into MADD as a "magic number," not a good practice. For a general purpose routine, you might tack the number of elements onto the calling SYS instruction as a third parameter, and use CHKCOM and FRMEVL to pick it up. The result would be left in the FAC in floating point, where it could be converted to integer format by calling AYINT.

The error flags OVRFLO and UNDFLO, associated with the ROM math routines, are then cleared. The pointer into the integer output table OUTBLE is reset, and we're ready for the main addition loop.

(Continued on next page)

In the addition loop, you'll notice that various registers have to be set up before calling the routines. For example, the accumulator must be set up with low byte, Y index register with the high byte of a variable's address, before calling MOVFM. In the case of MOVMF, it's the X and Y index registers.

The array pointers are then incremented by 5, the OUTBLE pointer is incremented by 2 (a Microsoft integer is 2, 8-bit bytes), the loop counter is decremented, and we do it again until 32 elements have been processed.

A BASIC test routine for MADD is shown in listing 2. Two 32 element arrays are set up, MADD is called, and the result is printed.

Incidentally, the assembly shown in listing 1 was done with the PAL assembler for the PET, written by Brad Templeton, the author of POWER. I recommend it highly.

Multi-Dimensional Arrays

The variables in the calling instruction need not be one-dimensional arrays. For example, suppose RL is 32×1 and DV is 32×5. The calling instruction

SYS XXX,RL(0),DV(0,3)

would have the effect of adding the 32 elements in the third (of five) columns of DV to the 32 elements in RL, leaving the result in RL. Notice that

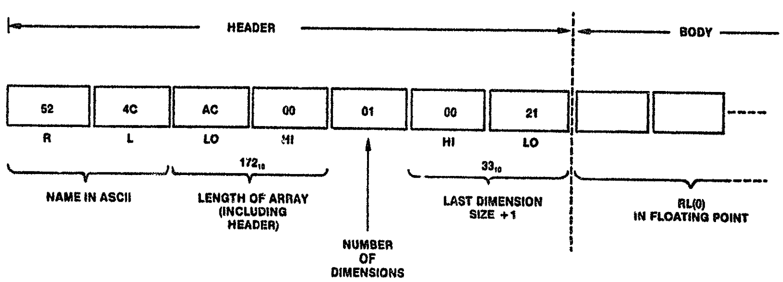
SYS XXX,RL(0),DV(3,0)

has a totally different and incorrect effect, because of the order of storage of the elements of DV.

Listing 1 *(continued)*

```
6006             outptr   *=    *+1       ; pointer into output table
                 ;
                 ; get pointers from sys statement
                 ; and set up in xxlo and xxhi
                 ;
6006 20 fd ae gtrl        jsr   chkcom    ; obtain address of rl(0)
6009 20 9e ad             jsr   frmevl
600c a5 47                lda   varpnt
600e 8d 00 60             sta   rllo
6011 a5 48                lda   varpnt+1
6013 8d 01 60             sta   rlhi
                 ;
6016 20 fd ae gtdv        jsr   chkcom    ; obtain address of dv(0)
6019 20 9e ad             jsr   frmevl
601c a5 47                lda   varpnt
601e 8d 02 60             sta   dvlo
6021 a5 48                lda   varpnt+1
6023 8d 03 60             sta   dvhi
                 ;
6026 a9 20                lda   #32       ; get the number of
6028 8d 04 60             sta   elemen    ; elements in the array
                 ;
602b a9 00                lda   #$0       ; clear error flags
602d 85 68                sta   ovrflo
602f 85 70                sta   undflo
                 ;
6031 a9 00                lda   #0        ; set up output pointer
6033 8d 05 60             sta   outptr    ;  to start of output table
                 ;
6036 ad 00 60 adloop      lda   rllo      ; main addition of rl(i) and dv(i)
6039 ac 01 60             ldy   rlhi
603c 20 a2 bb             jsr   movfm     ; rl(i) to fac
603f ad 02 60             lda   dvlo
6042 ac 03 60             ldy   dvhi
6045 20 8c ba             jsr   conupk    ; dv(i) to arg
                 ;
6048 20 6f b8             jsr   fadd      ; fac = fac + arg
                 ;
604b ae 00 60             ldx   rllo
604e ac 01 60             ldy   rlhi
6051 20 d4 bb             jsr   movmf     ; fac to rl(i)
                 ;
6054 20 bf b1             jsr   ayint     ; fac to integer
                 ;
6057 ac 05 60             ldy   outptr
605a a5 65                lda   faclo
605c 99 c8 60             sta   outble,y  ; integer to output table
605f ee 05 60             inc   outptr    ; increment output table pointer
6062 ee 05 60             inc   outptr    ; twice
                 ;
6065 18          incptr   clc             ; add 5 to loop pointers
6066 ad 00 60             lda   rllo      ; to move to next element
6069 69 05                adc   #5        ; in the array rl
606b 8d 00 60             sta   rllo
606e a9 00                lda   #0
6070 6d 01 60             adc   rlhi
6073 8d 01 60             sta   rlhi
                 ;
6076 18                   clc             ; ditto dv
6077 ad 02 60             lda   dvlo
607a 69 05                adc   #5
607c 8d 02 60             sta   dvlo
607f a9 00                lda   #0
6081 6d 03 60             adc   dvhi
6084 8d 03 60             sta   dvhi
                 ;
6087 ce 04 60             dec   elemen    ; test for complete
608a d0 aa                bne   adloop
                 ;
608c 60                   rts
```

***Figure 2:* One Dimensional Array Header**



Listing 2

```
10 REM MADDTEST ROUTINE
20 DIM RL(32),DV(32)
30 FOR J=0 TO 31
40 RL(J)=J:DV(J)=1.33
50 NEXT J
60 SYS 24582 ,RL(0),DV(0)
70 FOR J=0 TO 31
80 PRINT RL(J)
90 NEXT J
100 STOP
```

Peter Hiscocks is an instructor at Ryerson Polytechnical Institute, where he teaches courses in electronic and theatre technology. He builds computer interfaces on a freelance basis, and has just completed a computer-controlled sound system for the Royal Ontario Museum. You may contact him at Ryerson Polytechnical Institute, Electrical Department, 50 Gould St., Toronto, Canada M5B1E8.

MICRO

Bank-Switched JSR

by Terry M. Peterson

The Commodore SuperPET contains 96K of RAM — three times the memory of its plain sister, the CBM 8032. However, the extra 64K of RAM is "stacked" into 16 4K "banks" all addressed at $9000-$9FFF. Only one of these banks is accessible to the processor at any moment. The active bank is selected by setting the bank-select switch, a latch at $EFFC. For example, to activate bank seven, you would execute the equivalent of "POKE 61436,7". To use this memory with programs that won't fit within a single bank, you obviously need a method to jump from one bank to routines in other banks and back again — a "bank-switched JSR". Ideally, this bank-switched JSR would pass all processor registers between the called and calling routines just as does an ordinary JSR. Furthermore, it should place no restriction on the location of either routine within its bank. I have implemented such a bank-switched JSR using the 6502 "BRK" instruction together with a set of "jump" tables.

My bank-switched JSR sacrifices a small amount of memory, usually less than about one page per bank, in return for making inter-bank subroutine calls as easy as calls within a bank. To use this method, subroutine calls to other banks are assembled so they point to an entry in a jump table somewhere in the bank. The table consists of 8-byte entries, each of the following form:

```
EXTSUB BRK
       NN
       JSR XXXX
       BRK
       $FF
       RTS
```

where NN is the number of the bank containing the called routine and XXXX represents two arbitrary bytes. (For convenience in assembly, I usually point XXXX to EXTSUB — see listing 2.) Thus, at assembly time you need only the bank number of the called routine, not its actual address; you may assemble each bank's code independently. All banks' corresponding jump table entries must be at the same addresses, but you may place the tables anywhere in the bank — and there may be more than one table per bank. In operation, the system BRK vector is modified to point to the bank-managing routine shown in listing 1. This "bank manager" could be put in bank-switched memory, but it is more economical to put it in normal memory since you would need to put identical copies in all potentially active banks. Whenever EXTSUB in this example is called, the bank manager saves the current bank number on the machine stack and then changes the active bank number to NN. Bank NN contains the same jump table except that XXXX must point to the actual location of the called routine. Therefore, when the bank manager restores all the processor registers and returns control to the JSR following the byte "NN", the effect is (almost) the same as if a direct JSR were executing. (The difference is that three more bytes of stack space have been

This article shows a method for using the SuperPET's bank-switched memory with 6502 machine-language programs. Although written for the SuperPET, the technique may be used on any 6502 machine having a banked memory.

used.) Obviously, the only data we may pass to the called routine is in the processor registers — unless we first move the data to non-bank-switched RAM. When the called routine executes an RTS, the second BRK is executed and the "$FF" byte signals the bank manager that we wish to restore the previously active bank. The old bank number is pulled from the stack, then that bank is activated, and control is passed to the final RTS in the jump table entry with all processor register contents restored to their values at the time of the called routine's RTS.

Now that I've described *what* the bank manager does, let's look at listing 1 to see *how* it does it. First, it pushes on the stack (INDEX1), the contents of a pair of zero-page locations to be used temporarily. Next it fetches the program counter (PC) bytes from the stack where they were pushed during execution of the BRK. This is done using x-register indexed loads that take advantage of the fact that the stack pointer has already been loaded into the x-register by the SuperPET's interrupt-handling ROM routine before calling the bank manager. The PC is decremented by one and stored as (INDEX1). (INDEX1) now points to the NN byte of the jump table entry. NN is fetched and the original (INDEX1) is restored. Then NN is tested (TYA) to determine whether a jump (NN>=0) or return-jump (NN<0) is being requested. If it's a return, the branch to UNSWT is taken. Otherwise the top six stack items are lifted to make room for the current bank number below them. Then the new bank number is switched on and control is passed to IRQDON, the operating system routine that restores the processor registers from the stack and executes an RTI. At UNSWT the return-from-bank RTS is accomplished by retrieving the old bank number from under the top six stack items, putting it on top of the stack and moving the top items back down to undo the action of the jump-to-bank stack manipulation. Finally, IRQDON is called to resume normal program execution. Probably the most difficult part of the bank manager's action to visualize is its stack manipulation. The stack diagram shown in listing 1 should help.

Using this method of bank-switched JSR is fairly straightforward at the assember level, especially with an assembler that supports conditional assembly. An example of a source file containing code for bank-switched JSR's is shown in listing 2. This listing shows the source code describing the jump table in both the called and calling banks. The value of the label LCRBNK determines which bank's object code is assembled. Notice that routines may be called by the same name whether in-bank or out-of-bank, thanks to the conditional assembly feature.

Terry Peterson is engaged in photovoltaic cell research at Chevron Research Company. He may be contacted at 8628 Edgehill Ct., El Cerrito, CA 94530.

(Continued on next page)

Listing 1

```
0000            ; 6502 Assembly language
0000            ; code to manage bank switching on the SuperPET
0000            ;
0000            ; This code is entered via a BRK n. If 'n' is positive it
0000            ; is assumed to be the bank number to be made active. If
0000            ; 'n' is negative it flags a return to the calling program's
0000            ; bank number.
0000            ;
0000            ; as of 11/28/82
0000            ;
0000            ;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
0000            ;Stack Use:      BRK (+)                      BRK (-)
0000                            TO-BANK call              FROM-BANK call
0000            ;
0000            ;             At Entry ---> At Exit       At Entry -> At Exit
0000            ; Addrs,X     Contents      Contents      Contents  Contents
0000            ; 00FF         xx           new bank<--SP  xx       xx
0000            ; 0100         xx <--SP      YR            xx <--SP  old bank
0000            ; 0101         YR           XR            YR       YR <--SP
0000            ; 0102         XR           ACC           XR       YR
0000            ; 0103         ACC          SR            ACC      XR
0000            ; 0104         SR           PCL           SR       ACC
0000            ; 0105         PCL          PCH           PCL       SR
0000            ; 0106         PCH          old bank      PCH      PCL
0000            ; 0107                                    old bank  PCH
0000            ;
0000            ;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
0000            ;
0000                   * = 0              ;(For example assembly)
0000            INDEX1 = 0                ;Any convenient z-page dbl. byte
0000            IRQDON = $E600            ;Restore regs. and RTI
0000            BNKSW  = $EFFC            ;Latch for active bank
0000            CURBNK = $9002            ;My conventional location
0000            ;
0000  A5 00     BANKER LDA INDEX1         ;Save INDEX1 on stack
0002  48               PHA
0003  A5 01            LDA INDEX1+1
0005  48               PHA
0006  BC 06 01         LDY $0106,X
0009  BD 05 01         LDA $0105,X        ;Get desired bank #
000C  D0 01            BNE BNK100         ;  from (PC-1)
000E  88               DEY
000F  84 01     BNK100 STY INDEX1+1
0011  A8               TAY
0012  88               DEY
0013  84 00            STY INDEX1
0015  A0 00            LDY #0
0017  B1 00            LDA (INDEX1),Y     ;got it
0019  A8               TAY
001A  68               PLA                ;Restore (INDEX1)
001B  85 01            STA INDEX1+1
001D  68               PLA
001E  85 00            STA INDEX1
0020  98               TYA                ;a return-from-bank call?
0021  30 1E            BMI UNSWT          ;Yes
0023  9D               .BYT $9D,$FF,$00 ;STA $00FF,X (force abs. addr.
0024  FF
0025  00
0026  A0 06            LDY #6             ;Move six stack items up one
0028  68        BNK200 PLA                ;Get top item
0029  9D 00 01         STA $0100,X        ;Move up
002C  E8               INX                ;Next
002D  88               DEY                ;Done?
002E  D0 F8            BNE BNK200         ;No
0030  AD 02 90         LDA CURBNK         ;Current bank (by convention)
0033  48               PHA                ;Insert in stack
0034  8A               TXA                ;Calc. stack adj.
0035  38               SEC
0036  E9 08            SBC #8             ;Point to new bank item
0038  AA               TAX
0039  9A               TXS
003A  68               PLA                ;Get new bank
003B  8D FC EF         STA BNKSW          ;Turn it on
003E  4C 00 E6         JMP IRQDON         ;Go finish interrupt
0041  BD 07 01  UNSWT  LDA $0107,X        ;Get old bank #
0044  48               PHA                ;Put on stack
0045  A0 06            LDY #6             ;Shuffle 6 items down
0047  BD 06 01  UNSLOP LDA $0106,X
004A  9D 07 01         STA $0107,X
004D  CA               DEX                ;Next
004E  88               DEY                ;Done?
004F  D0 F6            BNE UNSLOP         ;No
0051  68               PLA                ;Get old bank
0052  8D FC EF         STA BNKSW          ;Reset it
0055  68               PLA                ;Adj. stack
0056  4C 00 E6         JMP IRQDON         ;Finish up
0059                   .END
```

Listing 2

```
; Example jump table for bank switching
;
;          as of 12/05/82
;
.IFN >LMABNK-LCRBNK
LOADTR >                   ;Dummy def. outside main bank
          BRK              ;LOADTR in bank LMABNK
          .BYTE LMABNK
          JSR LOADTR
          BRK
          .BYTE $FF        ;Back to caller
          RTS
```

NIBBLE EXPRESS III
YOUR APPLE* WILL LOVE IT!

TABLE OF CONTENTS

Turtle Graphics
Applesoft Line Cruncher
TEA: Income System
MAMA
Amper Free Space Catalog
Machine Language Editor
Apple CALLENDAR
Peaceful Coexistence
Apple Slugger
Reverse Key
Amper Find
Bond Manager
DOS 3.3 + 2
Amper Speed
Apple Darts
MicroCalc
Variable Cruncher
Life
Apple Flash
Checker
GO: Greeting Program
Quick Sort
Applesoft Line Editor
Othello
SPRINT
Disk Dump
Hi Res Colors
Apple Record Command System
Apple Art Gallery
Game I/O Expmt
Super Keypad
Disk Commander
Compare Applesoft
Appl. L Soft
Electronic Message Center
The Shape
Disk Map
Apple Bowl Football
Apple Scroller
Lower Case Letters
[illegible] Box
Disk Doctor
[illegible]
DOS Command Editor
[illegible] Driver

NOW AVAILABLE!

NIBBLE EXPRESS is an Anthology of the Major Articles and Programs appearing in Volume 3 of NIBBLE Magazine . . . nearly $400 worth of Programs for your Apple II, ACE, or other Applesoft-compatible computer!*

NIBBLE EXPRESS III contains *up-to-date enhancements, KeyPerfect Tables*, and the *Best of NIBBLE!*

Even if you have all 8 issues of NIBBLE in 1982, you'll want the Express to have updated program listings in one convenient package! It's a MUST for your Library!

NIBBLE EXPRESS III is a bargain at $17.95 plus $1.75 Postage/Handling ($2.50 outside the U.S.)

YOUR APPLE WILL LOVE IT!
(And so will you!)

NIBBLE
P.O. Box 325
Lincoln, MA 01773

Yes! I want to reserve NIBBLE EXPRESS Vol. III for my Library! Here's my ☐ Check ☐ Money Order for $17.95 plus $1.75 postage/handling. (Outside U.S. add $2.75 for postage/handling).

☐ Also send me NIBBLE EXPRESS Vol. I at $12.95 Plus postage/handling (see above).

☐ Send me NIBBLE EXPRESS VOL. II at $14.95 plus postage/handling (see above).

Master Card & Visa Accepted

Card # ____________________ Expires: __________

Signature: ________________________________

Telephone: ________________________________

Please Print Clearly

Name: ________________________________

Address: ________________________________

City ____________________ State ________ Zip ______

Your check or money order must accompany your order to qualify. Outside U.S.: Checks must be drawn on a U.S. Bank.

*Apple is a registered trademark of Apple Computer Inc.
*ACE is a trademark of Franklin Computer Co.

Directory Menu for the Commodore 64

by Al Korostynski

The Commodore directory can be accessed by opening it as a file and reading program and filenames. The directory can be read with GET# statements, but this method is rather slow: 50 programs take almost 20 seconds to display (it would be faster to list the directory the usual way, load the program, and run). I wrote a short machine-language routine to speed things up: 50 programs take about five seconds to display. The program stores the machine-language routine in the second cassette buffer and opens the directory after the drive number has been selected. Two strings, one for the header and one for the filenames, are pre-extended and filled with 17 spaces.

The machine-language program reads each directory entry and stores the filename in the pre-extended string. If the filename is a program, the "p" is stored in the 17th character of the string. BASIC regains control and stores the program name in an array. Then the machine-language routine gets the next filename. This looping continues until all the entries are read, then the directory is displayed.

The Directory Menu program can be stored as the first program on a disk. If you type 'LOAD"*",8' and "RUN" the program will execute. The other programs on the disk could be modified to chain back to the directory program at the end of their execution. Inexperienced users will find this arrangement especially useful because it allows them to run programs on a disk without having to learn and remember many disk commands or program names.

Be sure to save the program before using it for the first time. There are two reasons for this. First, you may make an error in entering the machine-language data statements, which could destroy

Listing 1

```
10 POKE53280,6:POKE53281,1
20 PRINT"■■"
100 DIM F$(224)
110 REM  -- POKE ML ROUTINE --
120 FOR X=880 TO 950: READ Z: POKE X,Z: CS=CS+Z: NEXT
130 IF CS<>10432 THEN PRINT "■■ERROR - CHECK DATA STATEMENTS...■": END
150 REM  -- READ DIRECTORY --
160 PRINT TAB(12)"■ DIRECTORY MENU ■■■"
170 PRINT "DRIVE #  0■■■";: INPUT D$
180 D=VAL(D$): IF D$<>"0" AND D$<>"1" THEN 160
190 OPEN15,8,15
200 PRINT#15,"I"+D$: GOSUB 290
210 OPEN1,8,0,"$"+D$: GOSUB 290
220 H$="                 "+"": SYS 880: REM  17 SPACES - READ HEADING
230 IF ST THEN CLOSE1: CLOSE15: GOTO 340
240 F$="                 "+"": SYS 880: REM  17 SPACES - READ FILENAME
250 IF RIGHT$(F$,1)="P" THEN C=C+1: F$(C)=LEFT$(F$,16)
260 GOTO 230
270 :
280 REM  -- DISK ERROR ROUTINE --
290 INPUT#15,EN,EM$,ET,ES: IF EN=0 THEN RETURN
300 PRINT EN;EM$;ET;ES
310 CLOSE1: CLOSE15: END
320 :
330 REM  -- DISPLAY MENU --
340 IF F=0 OR F=C THEN S=1: F=C: IF C>40 THEN F=40: GOTO 360
350 IF C>F THEN S=S+20: F=C: IF C>S+20 THEN F=S+19
360 L=0
370 PRINT "■■DRIVE #:■";D;"■DISK NAME:■ ";H$"■"
380 FOR X=S TO S+19
390 L=L+1
395 IFLEN(F$(X))=0THEN420
400 IFASC(F$(X))=32THEN420
410 PRINT "■"RIGHT$(STR$(X),2)"■";CHR$(95);" ";F$(X)
420 NEXT
430 PRINT
440 PRINT "■ ENTER: <RET> FOR NEXT PAGE   @ TO END ■"
450 PRINT TAB(9)"■ SELECT PROGRAM ■";:INPUTG$
500 IF G$="@" THEN PRINT "■":END
510 G=VAL(G$): IF G=0 THEN 340
550 REM  -- CHAIN --
560 F$=F$(G)
570 IF RIGHT$(F$,1)=" " THEN F$=LEFT$(F$,LEN(F$)-1): GOTO 570
580 PRINT "■LOAD"+CHR$(34)+D$+":"+F$+CHR$(34)+",8"
590 PRINT "■■■■?"+CHR$(34)+"■"+CHR$(34)+":RUN"
600 POKE 631,19:POKE 632,13:POKE 633,13:POKE 198,3
610 END
870 REM  -- ML ROUTINE DATA --
880 DATA 162,1,32,198,255,160,1,177,71,133
890 DATA 99,200,177,71,133,100,162,32,32,207
900 DATA 255,202,224,27,208,248,32,207,255,201
910 DATA 34,240,3,202,208,246,240,29,160,0
920 DATA 32,207,255,201,34,240,5,145,99,200
930 DATA 208,244,32,207,255,240,10,201,80,208
940 DATA 247,160,16,145,99,208,241,32,204,255
950 DATA 96
```

Listing 2

```
                      ************************************
                      *                                  *
                      *          MENU DIRECTORY          *
                      *          BY AL KOROSTYNSKI       *
                      *                                  *
                      ************************************
                                START
                      *       EQUATES
                      QUOTE     EQU  $22
                      VARPTR    GEQU $47
                      FPACC     GEQU $63
                      IOSET     EQU  $FFC6
                      IORSTOR   EQU  $FFCC
                      GETCHAR   EQU  $FFCF

                                ORG  $370

0370 A201             BEGIN     LDX  #$01
0372 20C6FF                     JSR  IOSET
0375 A001                       LDY  #$01
0377 B147                       LDA  (VARPTR),Y
0379 8563                       STA  FPACC
037B C8                         INY
037C B147                       LDA  (VARPTR),Y
037E 8564                       STA  FPACC+1
0380 A220                       LDX  #$20
0382 20CFFF           LOOP      JSR  GETCHAR
0385 CA                         DEX
0386 E01B                       CPX  #$1B
0388 D0F8                       BNE  LOOP
038A 20CFFF           AGAIN     JSR  GETCHAR
038D C922                       CMP  #QUOTE
038F F005                       BEQ  NEXT
0391 CA                         DEX
0392 D0F6                       BNE  AGAIN
0394 F01D                       BEQ  DONE
0396 A000             NEXT      LDY  #$00
0398 20CFFF           LOOP1     JSR  GETCHAR
039B C922                       CMP  #QUOTE
039D F005                       BEQ  AGAIN1
039F 9163                       STA  (FPACC),Y
03A1 C8                         INY
03A2 D0F4                       BNE  LOOP1
03A4 20CFFF           AGAIN1    JSR  GETCHAR
03A7 F00A                       BEQ  DONE
03A9 C950                       CMP  #$50
03AB D0F7                       BNE  AGAIN1
03AD A010                       LDY  #$10
03AF 9163                       STA  (FPACC),Y
03B1 D0F1                       BNE  AGAIN1
03B3 20CCFF           DONE      JSR  IORSTOR
03B6 60                         RTS

                                END
```

your work. Second, when the menu program is running, it is overwritten as you chain to your first program.

The machine-language routine loads into the second cassette buffer. The routine is fully relocatable, so if there is a conflict store it elsewhere.

To run the program, enter the drive number (0 or 1). The program takes a few seconds to read the directory and then it will display the program names on the disk. If there are more than 20 names, pressing return will list the next page. All pages may be viewed in this fashion and the pages will wrap around to the first page after the last page is displayed. To end the program, press "@". To make a selection, enter the appropriate number and press return.

Note that while the program will display only program names and not sequential or relative filenames, some files are stored as program file-types, such as Word-Pro, machine language, and other files. Care must be taken not to select them from the menu.

MICRO

Directory Menu for the Atari

by Jerry White

For a directory menu on the Atari all you need are your DOS master disk, a blank disk, and the two short BASIC programs that follow. If you don't remember a program name, you can give BASIC the command "DOS", then use option "A" to display your disk's directory file. DOS also provides a Binary Load option for loading machine-language programs. To load a BASIC program, it is a fairly simple matter to use the "B" option to get back to BASIC, then the BASIC "LOAD" or "RUN" commands.

Begin by inserting your Atari BASIC cartridge, then boot your computer with your DOS master disk in disk drive 1. At the "READY" prompt, type "DOS" and RETURN. Remove your DOS master and insert your blank disk. Type "I" and RETURN to tell DOS that you want to format a disk, then type "Y" and RETURN to format. Once you have a formatted disk, use DOS option "H" to write DOS files, then return to BASIC using DOS option "B".

When you boot using Atari's DOS 2.0S, the computer reads the DOS.SYS file first, then looks to see if there is a file named "AUTORUN.SYS" on the disk. If so, it loads AUTORUN.SYS and executes its instructions. To tell the computer to RUN a program called "MENU", type in the "MAKEAUTO" program. Before running any untested program, make sure you save it on a disk just in case something goes wrong.

Type in the MAKEAUTO program exactly as you see it printed, then save it using the immediate mode BASIC command SAVE"D:MAKEAUTO".

Now type "RUN" and RETURN to create your AUTORUN.SYS file. You will be instructed to insert your disk in drive 1, then press the START button to begin. If all went well, your AUTORUN.SYS file will be written onto the disk within a few seconds, and you will see the message BASIC IS READY.

Before you begin to type in the "MENU" program, type "NEW" and RETURN to remove the MAKEAUTO program from your computer's memory.

I used some of Atari's graphic characters and some inverse video letters in the MENU program. In line 140, the straight vertical line character is entered by holding the SHIFT key while typing the "=" or down-arrow key. The characters within quotes in line 280 are entered by holding the CTRL key while typing the "Q" key, the "R" key 33 times, then the "E" key. Similarly, the characters within quotes in line 550 are entered by holding the CTRL key while typing the "Z" key, the "R" key 33 times, then the "C" key.

Use inverse video to enter the word "RETURN" in line 290, the word "SELECTION" in line 310, and the words "NUMBER" and "RETURN" in line 540.

When you have finished typing in the MENU program, save it on your disk using the immediate mode command, SAVE"D:MENU".

Now go back to DOS and copy a few of your BASIC or machine-language programs onto this disk using DOS option "O". At this point, if you typed in the MAKEAUTO and MENU programs correctly and followed the preceeding instructions, you are ready to begin using your new automated system. Just turn off your computer and turn it on again. DOS.SYS should load, then turn over control to AUTORUN.SYS, which will run the MENU program.

MENU will display the contents of your disk (up to 34 files) excluding DUP.SYS and AUTORUN.SYS. All you have to do now is type your selection number and RETURN. Isn't automation wonderful?

Listing 1

```
1 REM MAKEAUTO by Jerry White
2 REM version date 4/12/83
10 GRAPHICS 0:POKE 82,2:POKE 83,39:POKE 752,1
20 ? :? "INSERT DESTINATION DISK IN DRIVE 1"
30 ? :? "PRESS START TO CREATE AUTORUN.SYS.":POKE 53279,8
40 IF PEEK(53279)<>6 THEN 40
50 ? :? "CREATING AUTORUN.SYS TO RUN MENU."
60 TRAP 300:CLOSE #1:OPEN #1,8,0,"D:AUTORUN.SYS"
70 TRAP 400:FOR BYTE=1 TO 148:READ IT:
   PUT #1,IT:POSITION 17,10:? BYTE;:NEXT BYTE
80 CLOSE #1:POKE 752,0:? :? :? "BASIC":? "IS";:END
100 DATA 255,255,0,6,133,6,162,0,189,26
110 DATA 3,201,69,240,5,232,232,232,208,244
120 DATA 232,142,105,6,189,26,3,133,205,169
130 DATA 107,157,26,3,232,189,26,3,133,206
140 DATA 169,6,157,26,3,160,0,162,16,177
150 DATA 205,153,107,6,200,202,208,247,169,67
160 DATA 141,111,6,169,6,141,112,6,169,10
170 DATA 141,106,6,96,172,106,6,48,9,185
180 DATA 123,6,206,106,6,160,1,96,138,72
190 DATA 174,105,6,165,205,157,26,3,232,165
200 DATA 206,157,26,3,104,170,169,155,160,1
210 DATA 96,0,0,0,0,0,0,0,0,0
220 DATA 0,0,0,0,0,76,0,0,0,34
230 DATA 85,78,69,77,58,68,34,78,85,82
240 DATA 255,255,226,2,227,2,0,6
300 ? :? :? "UNABLE TO OPEN AUTORUN.SYS FILE.":GOTO 80
400 ? :? :? "UNABLE TO READ BYTE #";BYTE;" OF 148."
410 ? "FIX PROGRAM DATA THEN RERUN.":GOTO 80
```

Listing 2

```
100 GOTO 480:REM MENU by Jerry White
110 REM version date 4/12/83
120 IF DREC$(4,8)=" FREE" THEN GOTO 240
130 NUM=NUM+1:WORK$=DREC$(3,10):WORK$(9,9)=".":
     WORK$(10,12)=DREC$(11,13)
140 LINE$="| ":LINE$(LEN(LINE$)+1)=WORK$:
     DISK$(LEN(DISK$)+1)=WORK$
150 LINE$(14,14)="(":JW=USR(1536,ADR(LINE$)):NUM$=STR$(NUM)
160 IF NUM<10 THEN NUM$(2,2)=NUM$(1,1):NUM$(1,1)="0"
170 RETURN
180 TRAP 280:INPUT #1,DREC$
190 IF DREC$(3,5)="DUP" OR DREC$(3,5)="AUT" THEN 180
200 GOSUB 120:? LINE$;NUM$;") ";
210 TRAP 240:INPUT #1,DREC$
220 IF DREC$(3,5)="DUP" OR DREC$(3,5)="AUT" THEN GOTO 210
230 GOSUB 120:? LINE$(2,14);NUM$;") | ":GOTO 180
240 IF NUM=0 THEN 280
250 POSITION 36,(NUM/2)+2:? "|"
260 POSITION 2,(NUM/2)+3:? "| "
270 POSITION 36,(NUM/2)+3:? "|"
280 ? "└─────────────────────────────────┘ "
290 ? "   TYPE 0 AND RETURN TO RESTART":CLOSE #1
300 POKE 752,1:POSITION 1,23:? "  ";:POSITION 10,22:? BLANK$
310 POSITION 1,22:POKE 752,0:? BLANK$(1,10);" SELECTION ";
320 POKE 764,255:TRAP 300:INPUT SEL:POKE 752,1:? " ";
330 IF SEL<>INT(SEL) THEN GOTO 300
3340 IF  NOT SEL THEN RUN
350 IF SEL=1 THEN WORK$=DISK$(1,12):GOTO 380
360 WORK$=DISK$((SEL-1)*12+1,(SEL-1)*12+12)
370 IF WORK$(10,10)=" " THEN WORK$=WORK$(1,8)
380 IF WORK$(1,4)="MENU" THEN RUN
390 IF WORK$(1,3)="DOS" THEN DOS
400 DREC$="":FOR STP=1 TO LEN(WORK$):
     IF WORK$(STP,STP)=" " THEN 420
410 DREC$(LEN(DREC$)+1)=WORK$(STP,STP)
420 NEXT STP:WORK$=DREC$:POSITION 17,22:? BLANK$(1,12)
430 DREC$="D:":DREC$(LEN(DREC$)+1)=WORK$
440 POSITION 1,22:? BLANK$(1,10);"LOADING ";WORK$:
     CLOSE #3:TRAP 450:RUN DREC$
450 TRAP 460:CLOSE #1:OPEN #1,4,0,DREC$:JW=USR(5576)
460 POSITION 4,22:? "I WAS UNABLE TO RUN THAT PROGRAM"
470 FOR JW=100 TO 255:SOUND 0,JW,10,JW/50:NEXT JW:RUN
480 GRAPHICS 0:CLOSE #3:OPEN #3,12,0,"S:":POKE 559,0
490 POKE 752,1:POKE 82,2:POKE 83,39:POKE 16,64:POKE 53774,112
500 POKE 710,160:POKE 709,13:POSITION 9,0:? "MENU by Jerry White"
510 DIM BLANK$(28):BLANK$=" ":BLANK$(28)=" ":BLANK$(2)=BLANK$
520 DIM WORK$(40),LINE$(20),DREC$(40),DISK$(400),NUM$(2)
530 CLOSE #1:OPEN #1,6,0,"D:*.*"
540 POSITION 4,2:? "TYPE PROGRAM NUMBER THEN RETURN"
550 POSITION 2,3:? "┌─────────────────────────────────┐ "
560 FOR ME=0 TO 24:READ IT:POKE 1536+ME,IT:
     NEXT ME:POKE 559,34:GOTO 180
570 DATA 104,104,133,206,104,133,205,160,0,177,205
580 DATA 201,32,208,4,169,46,145,205,200,192,13,208,241,96
```

Atari Graphics Notepad in FORTH

by Mike Dougherty

A Graphics Notepad allows the user to interactively construct images through simple keyboard commands. This article describes a minimal Graphics Notepad implemented on the Atari 800, written in APX fig-FORTH, V1.1.

A Graphics Notepad is a computer environment where graphic images may be interactively drawn under control of "English-like" commands typed on a "notepad." A typical implementation of a Graphics Notepad allows a video device (CRT or RF modulated television) to create two windows on the video screen. One window serves as the workspace for the graphic images; the second window forms the notepad. Commands written on the notepad are executed by the Graphics Notepad software. Any graphic results of the commands are displayed in the graphics workspace.

This article discusses a Graphics Notepad for the Atari 800. The Graphics Notepad implementation uses Atari graphics mode 8 (GR.8), APX fig-FORTH, and an Epson MX-80 F/T printer with GRAPHTRAX firmware. Since a Graphics Notepad environment is constantly expanding and growing with the user, the FORTH words defined in this article represent only a beginning. The ultimate capability of a Graphics Notepad is limited only by the imagination of the user.

Atari 800 Graphics Notepad Implementation

The Atari graphics modes 6, 7, and 8 are well-designed to implement a simple Graphics Notepad. This article will deal only with graphics mode 8 (GR.8), but the principles could be extended to cover the other graphics modes as well. GR.8 gives the user a graphic workspace of 160 dots vertically by 320 dots horizontally and creates a four-line notepad for communication with the Graphics Notepad software. The Atari Operating System (OS) currently allows the application software or language to plot points, draw lines, and fill areas of the graphic image. This small set of OS primitives relieves some of the software burden for high-level graphic commands.

The omission of high-level graphic commands may be corrected by software extensibility. The Graphics Notepad software or language should allow new primitives to be defined and easily used. This is where a language such as BASIC has difficulty. FORTH, on the other hand, is explicitly designed for extensibility and is used in this Graphics Notepad implementation. With FORTH, graphic images may be defined and used to build more complex images. While, in principle, BASIC can provide extensibility *via* subroutines, FORTH words are a far superior mechanism for software extension.

Finally, the Epson MX-80 printer is used to print the graphics workspace. This printer yields a 2.25-inch by 5.25-inch image, which is sufficient for the Atari implementation of the Graphics Notepad.

Software for Minimal Atari Graphics Notepad

The software capabilities of a minimal Graphics Notepad are a function of how the Graphics Notepad will be used. There is no clear distinction between software that is part of the minimal Graphics Notepad and software that is an extension. (This is the same problem that occurs when defining standard FORTH.) The words in Glossary 1, Screens 44 through 56, bring APX fig-FORTH to what I consider the minimal Graphics Notepad. The user commands are described next.

The FORTH word GTABLET initializes the Atari Graphics Notepad windows. GTABLET sets up the GR.8 screen, erases any previous results, and sets the colors. GTABLET may be executed at any time to clean the graphics workspace. The user exits the Graphics Notepad by executing the APX fig-FORTH word, XGR. (The APX graphics package must be LOADed prior to LOADing the Graphics Notepad.)

(Continued on page 68)

The FORTH word TEXT allows the Graphics Notepad user to write text directly from the keyboard onto the graphics workspace. TEXT uses an inverse video cursor under control of the arrow cursor keys for positioning the text. The cursor is moved horizontally, in steps of eight graphic dots (one full character width) and vertically, in single dot steps (one display line). The RETURN key may be used to move the cursor down by eight dots (one full character height). Once the cursor is properly positioned, the user adds the text to the graphics workspace by simply typing on the keyboard. The text added to the graphics workspace will overlap with any previous image value under the 8 x 8 character size. Upper-case, lower-case, and most graphic characters may be entered onto the graphics workspace. The user exits from the TEXT function by pressing the ESC key.

The FORTH word LABEL behaves identically to TEXT except that the text added to the graphics workspace will overwrite any previous image value under the 8 x 8 character size. TEXT is used to write *on top* of the graphics workspace, while LABEL is used to *replace* a part of the workspace with character text.

Both TEXT and LABEL use the Atari character generator ROM, beginning at memory address $E000 (decimal address 57344). This ROM defines each row of a character in eight consecutive bytes for a total character size of 8 x 8 dots. The word ASCII-> ROM translates the ASCII character into the beginning ROM address for that character. Since each GR.8 line is 40 bytes long, a character is put on the GR.8 screen by moving eight consecutive ROM character bytes to GR.8 memory at 40-byte intervals (separating the character bytes by 40-byte positions, each byte under the previous byte).

Both TEXT and LABEL make use of the CASE statement defined by Dr. C. E. Eaker. The original CASE FORTH code was presented in FORTH Dimension, Volume II, Number 3, pp. 37-40. Only the CASE-word names were changed when used in the Atari Graphics Notepad (Screen 52).

The FORTH word SAVE-SCREEN allows the user to save an intermediate version of the graphics workspace. Thus, if a command does not work as anticipated, the previously saved graphics workspace may be recovered by the FORTH word RESTORE-SCREEN. Note that these two words require a workspace buffer, WORKSPACE-BUF, of 6400 bytes in size. **Atari Systems with limited memory should not compile and use Screen 56.**

The FORTH word BORDER simply adds a line border around the graphics workspace. This command is used primarily with the hardcopy function to define the graphics workspace on the printed page.

Hardcopy Function

The Hardcopy function of this Graphics Notepad implementation is performed by an Epson MX-80 F/T printer with GRAPHTRAX firmware. APX fig-FORTH V1.1 opens channel #6 to the printer for use with the words PON and POFF. The Atari Graphics Notepad makes use of this channel with the word PRINT-CHAR. Any changes made in updates of APX fig-FORTH will necessitate changes in PRINT-CHAR. Glossary 2, Screens 31 through 43, define the FORTH hardcopy words used in the Atari Graphics Notepad.

The Epson printer gives high-resolution graphic output by allowing the user to control each dot on the page. For the normal graphics print mode, each 8-inch line contains 480 dots (1/60th inch per dot) and each 10-inch page contains 720 lines (1/72nd inch per line). In the super graphics print mode, the horizontal dots in each line overlap, yielding 960 dots per 8-inch line, producing a generally more pleasing image. The graphics print mode is user-selected by the two FORTH words NORMAL and SUPER. The super graphics mode prints each image pixel (dot) twice to eliminate the horizontal spacing between dots. Since the vertical spacing is 1/72nd of an inch, the super graphics printed image usually has a more solid appearance.

The vertical line spacing of each print line is adjustable from a single dot to an 85-dot spacing, allowing printed lines to be overlapped or widely separated. The Atari Graphics Notepad uses two vertical line spacings with the FORTH words FAST-GRAPH and SLOW-GRAPH. FAST-GRAPH prints graphic lines utilizing the full available column of eight dots and a line spacing of eight. SLOW-GRAPH uses a line spacing of only four and prints only the upper four dots of each graphic character. As a result, FAST-GRAPH will print the graphics workspace in 20 full lines (160 rows/8 dots per row), while SLOW-GRAPH takes 40 full lines (160 rows/4 dots per row).

SLOW-GRAPH is included in the Graphics Notepad due to two problems with the Atari 850 Interface Module firmware. While the Epson printer is in normal or super graphics print mode it does not interpret the graphic characters as having any special meaning. Each graphic character is simply a bit pattern telling a specific printer dot to print (1) or not (0). Unfortunately, the Atari 850 firmware does not have a graphics mode. Because of this, the graphics character 155 ($9B = 10011011) is interpreted as an end-of-line character, EOL. The 850, not knowing about the Epson graphics modes, decides that a printer requires a carriage return (CR), 13 ($0D = 00001101), instead of the EOL and automatically converts the EOL to a CR. From the viewpoint of a graphic character, this 155 to 13 conversion changes four dot values. FAST-GRAPH solves this problem by changing all 155 characters to 147 ($93 = 10010011) before sending the character to the Atari 850 for printing. The 147 is incorrect by only one bit, which usually will not be missed.

The second problem occurs when two consecutive carriage return characters are sent to the 850. Apparently, the Atari printers do not like consecutive CRs, so the 850 automatically inserts an extra space character, decimal 32 ($20 = 00100000), between the CRs. This can lead to very unusual results that are hard to correct on the Epson! Since the most common occurrence of two CRs occurs when using super graphics print mode (each character is printed twice), PRINT-DOTS converts the first 13 to a 9 ($09 = 00001001). The 9 is again incorrect by only one dot, which should never be missed. However, SLOW-GRAPH, which prints only the upper four bits of each line, will never be printing the 155 (EOL) or 13 (CR) characters. Thus, for cases when the hardcopy must be an exact duplicate of the graphics workspace (or all else fails), use SLOW-GRAPH for printing. Most of the time, FAST-GRAPH will be sufficient and will print in half the time.

There is one additional printer problem. The data link from the Atari 800 to the printer is a one-way link. Once data is sent to the printer, there is no guarantee that it reaches the printer without an error. I have found that with my six-foot ribbon cable, the data is occasionally susceptible to error. For normal listings, an error usually means a misspelled word. However, when transmitting graphics data, many strange things can, and do, happen! Short of improving the cable, do the following for a run-away Epson:

- Abort the FAST-GRAPH or SLOW-GRAPH by pressing any key (other than break).
- Turn off the EPSON to initialize some of the printer variables. (If the run-away printer is ejecting paper, do this first!)
- Manually position the paper as needed since most disasters include creative paper spacing.
- Turn on the printer.
- Software reset the printer with the FORTH word RESET-EPSON.

Although not the best solution, the above steps seem to work after enough tries.

Atari Graphics Notepad Applications

To illustrate a simple use of the Graphics Notepad, figures 1 through 5 were drawn with the Atari Graphics Notepad and printed on the Epson. The basic construction of each figure is discussed below. Except where noted, all hardcopy used the super graphic print mode and FAST-GRAPH printing.

Figure 1 is a demonstration of what TEXT and LABEL can do with interactive text entry. The image was created by using the Atari 800 keyboard alone.

Figures 2 through 5 illustrate the capability of FORTH to build upon images. The image word FAN was defined as:

```
: FAN                   ( x y ---)
  320 10 DO             ( Across the screen)
    2DUP                ( Save a copy of x,y)
    1 ROT ROT PLOT      ( Plot the start)
    1 I 0 DRAW          ( Draw a line of FAN)
  20 +LOOP
  2DROP ;               ( Clean up stack)
```

Figure 2 was created by the commands:

```
160 159 FAN BORDER SUPER FAST-GRAPH
```

The word FAN-LINE was defined, based upon FAN:

```
: FAN-LINE              ( x ---)
  160 39 DO             ( Down the screen)
    DUP I FAN           ( Draw a FAN)
  40 +LOOP              ( Every 40 lines down)
  DROP ;                ( Clean up column number)
```

Figure 3 was created by the commands:

```
0 FAN-LINE
319 FAN-LINE
160 120 FAN
BORDER SUPER FAST-GRAPH
```

The lettering was added with the TEXT and LABEL Graphics Notepad words. For figure 4, FAN-LINE was modified:

```
: FAN-LINE
  160 120 DO
    DUP I FAN
  5 +LOOP
  DROP ;
```

The Graphics Notepad "160 FAN-LINE" command creates the figure 4 ghostly specter with small beady eyes! Figure 5 is the same but with the NORMAL graphics print mode.

Hopefully the figures will inspire you to build your own Graphics Notepad and to create new and better graphic worlds. At the worst, you will end up having a lot of fun!

The author may be contacted at 7659 West Fremont Ave., Littleton CO, 80123.

Figure 1

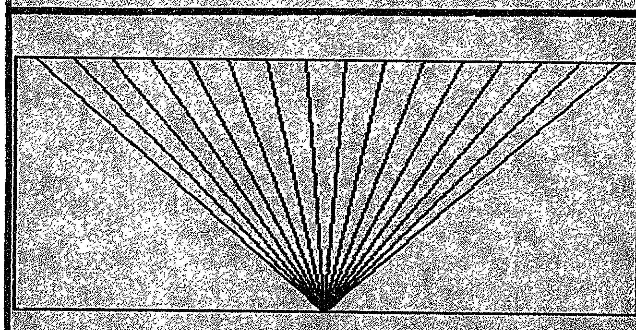
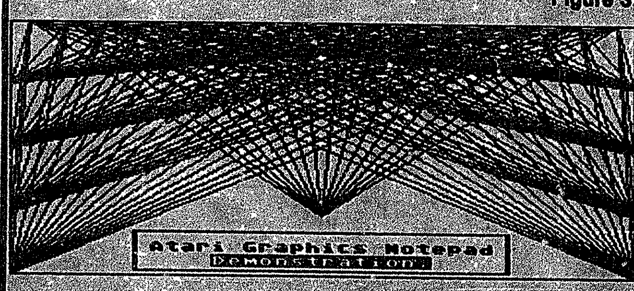


Figure 2

Figure 3



(Continued on page 71)

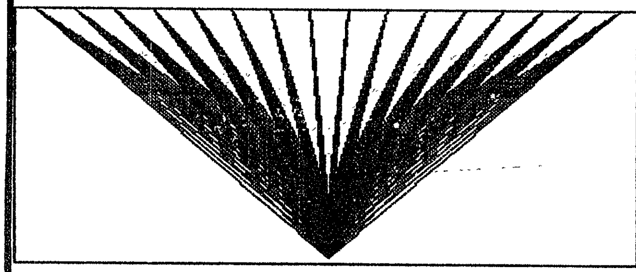


Figure 4

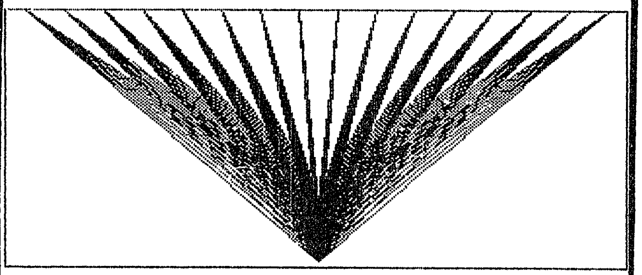


Figure 5

***Glossary 1.* Minimal Atari Graphics Notepad**

ASCII->ROM (ascii — ROMaddr) .... Screen 46
ASCII->ROM takes the "ascii" character and returns the Atari character generator ROM address, "ROMaddr", for that character's GR.8 pattern. The flag INVERSE is set to 128 if the character was in inverse mode.

BORDER (—) .... Screen 55
BORDER draws a single line border around the graphics workspace.

C-DOWN (—) .... Screen 49
Move the TEXT/LABEL cursor down one display line.

C-LEFT (—) .... Screen 48
Move the TEXT/LABEL cursor left eight dots, one character width.

C-RETURN (—) .... Screen 50
Move the TEXT/LABEL cursor down eight lines, one character height. The cursor horizontal position is not changed.

C-RIGHT (—) .... Screen 48
Move the TEXT/LABEL cursor right eight dots, one character width.

CURSOR (—) .... Screen 47
Toggel the current TEXT/LABEL cursor by inverting the eight graphic bytes representing the cursor positon. This routine is used to set or clear the current cursor position.

CURSOR-ADDR (— addr) .... Screen 45
Compute the beginning memory address, "addr", of the TEXT/LABEL cursor in the graphics workspace.

GNOTEPAD (—) .... Screen 44
Initialize the Graphics Notepad. This word may also be used to erase the current graphics workspace.

LABEL (—) .... Screen 54
Allow the user to replace portions of the graphics workspace with text entered from the keyboard. Exit from LABEL with the ESC key.

RESTORE-SCREEN (—) .... Screen 56
Restore the previously saved graphics workspace (re-quires a 6400 byte buffer).

SAVE-SCREEN (—) .... Screen 56
Save the current graphics workspace in a 6400 byte memory buffer.

SHOW (ascii —) .... Screen 51
Place the "ascii" character on the GR.8 graphics workspace at the current TEXT/LABEL cursor position. The cursor is moved one character to the right.

TEXT (—) .... Screen 53
Allow the user to write on top of the image in the graphics workspace with text entered directly from the keyboard. Exit from TEXT with the ESC key.

***Glossary 2.* Epson MX-80 F/T Hardcopy**

2**N (n — 2**n) .... Screen 33
Raise 2 to the "nth" power by multiplication — change to a CODE level word with shift instructions if 2**N is too slow.

850-KLUDGE (char — char) .... Screen 40
Correct for the actions of the 850 Interface Module on the print stream — refer to the article for details.

BIT-SLICE (bufaddr — bitslice) .... Screen 38
Accumulate bit7 (most significant bit) of the 8 bytes starting at "bufaddr". This "bit slice" is used as the vertical print column in the Epson graphics print mode. The top of the vertical column is stored in bit7 of the bit slice, while the bottom dot is bit0. The 8 bytes starting at "bufaddr" are shifted left one bit for the next call to BIT-SLICE.

EPSON-GRAPHICS (n —) .... Screen 35
Command the Epson to accept the next "n" bytes as graphic characters. Each graphic character will control the 8 vertical printing pins of the Epson, the top being bit7 and the bottom pin bit0.

FAST-GRAPH (—) .... Screen 43
Print the graphics workspace on the Epson printer, eight display lines at a time. Due to the 850 Interface Module, the EOL (155) and CR (13) characters may be modified — refer to the article for details.

GET-DATA (memaddr —) .... Screen 37
Move a vertical slice of 8 bytes from GR.8 memory to the buffer GR8-DATA for use by BIT-SLICE.

INIT-LINE (—) .... Screen 39
Initialize the Epson to receive graphics mode characters for a print line of the graphics workspace.

NORMAL (—) .... Screen 31
Set the GMODE flag for normal graphics print mode, 480 dots per 8 inch line.

NORMAL-EPSON (—) .... Screen 36
Reset the Epson line spacing to 1/6th inch default.

PRINT-CHAR (char —) .... Screen 32
Output "char" to the printer using printer channel #6 (PRINTER-CHANNEL) which was opened by APX fig-FORTH.

PRINT-CR (—) .... Screen 34
Print a carriage return on the Epson.

PRINT-CRS (n —) .... Screen 34
Print "n" carriage returns on the Epson.

PRINT-DOTS (addr mask — addr + 40) .... Screen 41
Print 1 to 8 graphics workspace display lines on the Epson printer. "Addr" points to the start of the graphics workspace lines in memory. The AND "mask" is used to determine which bits will be allowed to print. For FAST-GRAPH, "mask" is 255 ($FF) while for SLOW-GRAPH, "mask" is 240 ($F0).

PRINT-SPACE (—) .... Screen 34
Print a space on the Epson.

PRINT-SPACES (n —) .... Screen 34
Print "n" spaces on the Epson.

RESET-EPSON (—) .... Screen 36
Perform a software reset of the Epson printer for "run-away" situations due to noise glitches, etc.

SCREEN (— gr8base) .... Screen 36
Return the beginning memory address of the graphics workspace, "gr8base". This word is used in both the Epson hardcopy and minimal Graphics Notepad application.

SCREEN-DUMP (n —) .... Screen 42
Print the contents of the graphic workspace, "n" display lines at a time.

SETUP-EPSON (n —) .... Screen 39
Set the Epson print line to "n" rows.

SLOW-GRAPH (—) .... Screen 43
Print the graphics workspace on the Epson printer, four display lines at a time.

SUPER (—) .... Screen 43
Set the GMODE flag to super graphics print mode, 960 dots per 8 inch line. This mode allows each pixel (dot) to be printed twice and overlapped, forming a more solid image.

The Atari Graphics Notepad requires:

Atari 400/800
Atari 810 disk drives
32K bytes of memory.
APX fig-FORTH V1.1, Rev 1

```
SCR # 30
( Atari Graphics Notepad for APX fig-FORTH V1.1, Rev 1            )
(                                                                  )
(            by Mike Dougherty                                     )
(                                                                  )
( To LOAD both the Epson Hardcopy and the Graphics Notepad:        )
(   -- 30 LOAD                                                     )
(                                                                  )
( To LOAD the Atari Graphics Notepad only:                         )
(   -- remove comments on SCREEN definition, Forth Screen #44      )
(   -- 44 LOAD                                                     )
(                                                                  )
( To LOAD SAVE-SCREEN and RESTORE-SCREEN [requires 6400 bytes]:)
(   -- 56 LOAD                                                     )

-->
```

```
SCR # 31
( EPSON MX-80 F/T HARDCOPY FOR PRINTERS WITH GRAFTRAX[TM] )

6  CONSTANT PRINTER-CHANNEL     ( APX fig-Forth printer chan )
27 CONSTANT ASCII-ESC           ( For Epson printer commands )
0  VARIABLE GRB-DATA 6 ALLOT    ( For col of "vertical" bytes )
0  VARIABLE GMODE               ( Super/normal graphics mode )


: NORMAL                        ( --- )
    0 GMODE ! ;                 ( Set hardcopy normal graphics )

: SUPER                         ( --- )
    1 GMODE ! ;                 ( Set hardcopy super graphics )

-->
```

```
SCR # 32
( OUTPUT TOP OF STACK TO PRINTER )

: PRINT-CHAR                    ( char --- )
    PRINTER-CHANNEL             ( APX fig-Forth printer chan )
    PUTC                        ( Put a character )
    DROP ;                      ( Drop the status )

-->
```

```
SCR # 33
( RAISE 2 TO THE TOP OF THE STACK POWER )

: 2**N                          ( n --- 2**n )
    1 SWAP                      ( Initialize for 2**0 )
    -DUP IF                     ( If non-zero ... )
      0 DO                      ( For n times )
        2 *                     ( Raise by a power of 2 )
      LOOP
    ENDIF ;

-->
```

```
SCR # 34
( BASIC EPSON LINES )

: PRINT-CR                      ( --- )
    155 PRINT-CHAR ;

: PRINT-CRS                     ( n --- )
    0 DO  PRINT-CR  LOOP ;

: PRINT-SPACE                   ( --- )
    32 PRINT-CHAR ;

: PRINT-SPACES                  ( n --- )
    0 DO  PRINT-SPACE  LOOP ;

-->
```

```
SCR # 35
( SET EPSON INTO GRAPHICS MODE )

: EPSON-GRAPHICS                ( n --- )
    -DUP IF                     ( If any graphics bytes )
      ASCII-ESC PRINT-CHAR      ( Then init to expect graphics )
      GMODE @ IF                ( If super graphics mode )
        ASCII L PRINT-CHAR      (  use "<ESC>Lij" )
      ELSE                      ( Else normal graphics mode )
        ASCII K PRINT-CHAR      (  use "<ESC>Kij" )
      ENDIF
      256 /MOD SWAP             ( Where i=LSB, j=MSB of n )
      PRINT-CHAR                ( Note: i,j should not be 155 )
      PRINT-CHAR                (  "      or 13 due to 850 Inter-)
    ENDIF ;                     (         face Module )

-->
```

```
SCR # 36
( RESET FROM GRAPHICS PRINTING )

: NORMAL-EPSON                  ( --- )
    ASCII-ESC PRINT-CHAR        ( Reset the line spacing )
    ASCII 2   PRINT-CHAR ;      ( with "<ESC>2" )

: RESET-EPSON                   ( --- )
    ASCII-ESC PRINT-CHAR        ( Reset the entire Epson )
    ASCII @   PRINT-CHAR ;      ( with "<ESC>@" )

: SCREEN                        ( --- gr8base )
    106 C@                      ( Get current top of memory )
    256 *                       ( Form 16 bit address )
    7856 - ;                    ( Backup to first of GR.8 mem )

-->
```

```
SCR # 37
( MOVE DATA TO 8 BYTE BUFFER )

: GET-DATA                      ( memaddr --- )
    8 0 DO                      ( For 8 bytes of vert slice )
      DUP C@                    ( Get a byte )
      GRB-DATA I +              ( Buffer addr )
      C!                        ( Save whole byte )
      40 +                      ( Next line, 40 bytes down )
    LOOP
    DROP ;                      ( Clean up memaddr )

-->
```

```
SCR # 38
( ACCUM BIT7, SHIFT BUFFER )

CODE BIT-SLICE                  ( bufaddr --- bitslice )
  1 # LDA,  SETUP JSR,          ( Move addr to N,N+1 )
  BEGIN,                        ( For all 8 bits )
    CLC,                        ( To clear buffer )
    N )Y LDA,                   ( Get buffer byte )
    ROL.A,                      ( Put bit7 into carry )
    N )Y STA,                   ( Save shifted result )
    N 2 + ROL,                  ( Put carry into result )
    INY,                        ( Next buffer byte )
    8 # CPY,                    ( All 8 ? )
  0= END,                       ( Not yet )
  0 # LDY,                      ( Yes, reset y reg )
  N 2 + LDA,  PUSHOA JMP,       ( Exit with N+2 on stack )
FORTH -->
```

```
SCR # 39
( INITIALIZE )

: SETUP-EPSON                     ( n --- )
    10 PRINT-CRS                  ( Space down the paper )
    ASCII-ESC PRINT-CHAR          ( Set line spacing to n dots )
    ASCII A   PRINT-CHAR          ( with "<ESC>An" )
    PRINT-CHAR ;

: INIT-LINE                       ( --- )
    13 PRINT-SPACES               ( Space over the paper )
    GMODE @ IF                    ( If super graphics mode )
      640 EPSON-GRAPHICS          ( Epson will get 640 char/line )
    ELSE                          ( Normal graphics mode )
      320 EPSON-GRAPHICS          ( Epson will get 320 char/line )
    ENDIF ;
-->
```

```
SCR # 40
( GRAPHICS KLUDGE DUE TO 850 INTERFACE MODULE FIRMWARE )

: 850-KLUDGE                      ( char --- char )
    DUP 155 = IF                  ( If char is Atari EOL )
      DROP 147                    ( Convert, else 850 makes CR )
    ENDIF                         ( 147 is better than 13 )
    GMODE @ IF                    ( If in super graphics mode )
      DUP                         ( Each character printed twice )
      DUP 13 = IF                 ( But 2 CRs, 850 inserts space )
        DROP 9                    ( So conver CR to 00001001 )
      ENDIF                       ( 9 is close, no added $20 )
      PRINT-CHAR                  ( Output char twice )
    ENDIF ;

-->
```

```
SCR # 41
( PRINT A LINE OF DOTS )

: PRINT-DOTS                      ( addr mask --- addr+40 )
    INIT-LINE                     ( Set up line spacing, graph )
    40 0 DO                       ( For 40 bytes of display mem )
      OVER GET-DATA               ( Get 8 columns of data )
      8 0 DO                      ( For dots in each column )
        GR8-DATA BIT-SLICE        ( Get the vertical bit slice )
        OVER AND                  ( Keep mask bits of data )
        850-KLUDGE                ( Correct for 850, do GMODE )
        PRINT-CHAR                ( Output graphics column )
      LOOP                        ( All 8 vertical columns )
      SWAP 1+ SWAP                ( Next 8 columns )
    LOOP                          ( All across the screen row )
    DROP                          ( Drop mask, leave addr+40 )
    PRINT-CR ;        -->         ( Move to a new line )
```

```
SCR # 42
( DUMP A GRAPHICS 8 SCREEN )
: SCREEN-DUMP                     ( n --- )
    DUP SETUP-EPSON               ( Set Epson to n dots/line )
    SCREEN                        ( Get the start of GR.8 mem )
    OVER 160 SWAP /               ( # of n dot lines to print )
    0 DO                          ( For each line of n dots )
      OVER MINUS 8 + 2**N         ( Form bitmask for n dots )
      MINUS 256 +                 ( as 256 - 2**[8-n] )
      PRINT-DOTS                  ( Print the dots not masked )
      OVER 1 - 40 * +             ( Update GR.8 address )
      ?TERMINAL IF                ( If any key is pressed )
        KEY DROP LEAVE            ( Get/ignore key, abort loop )
      ENDIF
    LOOP                          ( For all n dot lines )
    2DROP                         ( Clean up n,addr )
    NORMAL-EPSON ;    -->         ( Reset printer to normal mode )
```

```
SCR # 43
( BASIC SCREEN DUMP MODES )

: FAST-GRAPH                      ( --- )
    8 SCREEN-DUMP ;               ( Print GR8 screen 8 rows/line )

: SLOW-GRAPH                      ( --- )
    4 SCREEN-DUMP ;               ( Print GR8 screen 4 rows/line )

-->
```

```
SCR # 44
( MINIMAL GRAPHICS TABLET FOR ATARI )

: GNOTEPAD                        ( --- )
    8 GR.                         ( Initialize mode*8 graphics )
    SCREEN 6400 ERASE             ( Erase the graphics workspace )
    1 0 0 SETCOLOR                ( Set the video colors )
    2 9 3 SETCOLOR                ( Change according to personal )
    4 6 1 SETCOLOR ;              ( taste. )


( : SCREEN         )              ( Remove comments if LOADing   )
(     106 C@       )              ( The Atari Graphics Notepad   )
(     256 *        )              ( only --- else leave commented )
(     7856 - ;     )              ( out of the Forth LOAD )

-->
```

```
SCR # 45
( PUT CHARACTERS ONTO GR.8 SCREEN USING ATARI CHAR ROM )

0 VARIABLE CURSOR-X               ( X position on screen )
0 VARIABLE CURSOR-Y               ( Y position on screen )
0 VARIABLE INVERSE                ( Inverse character flag )
0 VARIABLE TEXT/LABEL             ( Flag to use different modes )

: CURSOR-ADDR                     ( --- addr )
    SCREEN                        ( Get base addr of screen )
    CURSOR-Y @ 40 * +             ( Add for line position )
    CURSOR-X @ + ;                ( Add for col  position )

-->
```

```
SCR # 46
( ASCII VALUE TO CHARACTER GENERATOR ROM ADDRESS )

57344 CONSTANT CHAR-ROM           ( Atari char ROM at $E000 )

: ASCII->ROM                      ( key --- addr )
    DUP 128 AND INVERSE !         ( Isolate inverse flag )
    127 AND                       ( Strip any inverse value )
    DUP 32 < IF                   ( If control character )
      64 +                        ( Map into 64-95 )
    ELSE                          ( Not control character )
      DUP 96 < IF                 ( If uppercase )
        32 -                      ( Map into 0-63 )
      ENDIF                       ( Else leave value 96-127)
    ENDIF
    8 *                           ( 8 bytes per char )
    CHAR-ROM + ;          -->     ( Add base address )
```

```
SCR # 47
( TOGGEL TEXT/LABEL CURSOR ON/OFF )

: CURSOR                          ( --- )
    8 0 DO                        ( For all 8 lines*of char )
      CURSOR-ADDR I 40 * +        ( Get address of graph cursor )
      DUP C@                      ( Get graphics byte )
      255 XOR                     ( Invert value )
      SWAP C!                     ( Set back in memory )
    LOOP ;

-->
```

```
SCR # 48
( TEXT/LABEL CURSOR MOVEMENT )

: C-RIGHT                         ( --- )
    CURSOR                        ( Toggel cursor to original )
    CURSOR-X @ 1 +                ( Move to the right 1 char )
    DUP 39 > IF DROP 0 ENDIF      ( Wraparound if necessary )
    CURSOR-X !                    ( Save new position )
    CURSOR ;                      ( Set the new cursor position )

: C-LEFT                          ( --- )
    CURSOR                        ( Toggel cursor to original )
    CURSOR-X @ 1 -                ( Move to the left 1 char )
    DUP 0 < IF DROP 39 ENDIF      ( Wraparound if necessary )
    CURSOR-X !                    ( Save new position )
    CURSOR ;                      ( Set the new cursor position )
-->
```

MICRObits

The Cheap Assembler includes unlimited length labels, free field programming, two-pass RAM/disk-based asssembly, ten-command text editor, interactive operation, tutorial manual, and demonstration routines. APPLE II+, 48K, DOS 3.3 required. Send $20.00 plus $4.00 P&H to: Thunder Software, P.O. Box 31501, Houston, TX 77231, (713) 728-5501.

Payroll Compatible with 64K Apple computers. Features: 100 employees per disk, time-card adding, automatic overtime, user-changeable tax tables, five miscellaneous deductions — fixed or percentage amounts, tips, automatic printing of checks, W-2 forms, and reports. Bill Martin, 6485 Diana Drive #2, Poland, OH 44514, (216) 757-2143.

Digitizer/Graphics Tablet THE HELPING HAND — more than just a drawing device. More versatile than paddles or joysticks. Drawing board is 16" × 20" and includes two function keys. Connects to the 9-pin game port of VIC-20, C64, Atari, TI, etc. $44.50. Persimmon Peripherals, Route 2, Box 2306A-MI, Claytin, GA 30525.

Peripherals for the VIC-20 and C64 Light pen model BR2064, very accurate readings in vertical and horizontal, push-button switch, excellent documentation and software listings included. $26.95. 51K plus expansion memory expands VIC-20 to 56K RAM, assembled and tested. Excellent documentation. $140.00. Miami Valley Micro Systems, 3341 Sheffield Rd., West Carrollton, OH 45449.

OSI — Affordable DBM B&W File Master runs under OS65D V3.3. 32K RAM minimum. Features: user and/or predefined files w/codes, formatted viewing and inputting, find, edit, delete, and page. Multiple dumps. Only $55.00. Label printer add $25.00, report generator add $25.00, manual only $10.00. For more information contact: Bunin & Ward Computer Services, P.O. Box 895, Church Street Sta., New York, NY 10008, (212) 434-5760.

VIC-20 Plug-ins Reset Switch, uses the serial port — $5.00. Amplifier Adapter Cable, uses the audio/video port — $10.00. Send check or money order to: DLW Co., Box 485, Cupertino, CA 95015.

Attention Software Vendors Expand your market! Your BASIC programs, 8K or smaller, will run on OSI C1P, or C4P. I will translate your program for either/both machines on a fee basis. Bruce Showalter, 857 Cedar, Abilene, TX 79601.

Apple Software Learn Applesoft programming, use of the disk drive, graphics, etc. Each learning package includes two disks and instructional tutorial manual. The Apple's Core — $49.95. Part II: The Seed — $59.95. *Free brochure*. The Professor, P.O. Box 301M, Swanton, VT 05488, (514) 747-9130.

MICRO

Atari

```
SCR # 49
( TEXT/LABEL CURSOR MOVEMENT )

: C-UP                                  ( --- )
    CURSOR                              ( Toggel cursor to original )
    CURSOR-Y @ 1 -                      ( Move up one display line )
    DUP 0 < IF DROP 152 ENDIF           ( Wraparound if necessary )
    CURSOR-Y !                          ( Save the new position )
    CURSOR ;                            ( Set the new cursor position )

: C-DOWN                                ( --- )
    CURSOR                              ( Toggel cursor to original )
    CURSOR-Y @ 1 +                      ( Move down one display line )
    DUP 152 > IF DROP 0 ENDIF           ( Wraparound if necessary )
    CURSOR-Y !                          ( Save the new position )
    CURSOR ;                            ( Set the new cursor position )
-->
```

```
SCR # 50
( TEXT/LABEL CURSOR MOVEMENT )

: C-RETURN                              ( --- )
    CURSOR                              ( Turn off current cursor )
    CURSOR-Y @ 8 +                      ( Move down 8 rows )
    DUP 152 > IF DROP 0 ENDIF           ( Wrap around )
    CURSOR-Y !                          ( Save new value )
    CURSOR ;                            ( Turn on current cursor )

-->
```

```
SCR # 51
( PUT ATARI CHAR ON GRAPHICS 8 SCREEN )
: SHOW                                  ( ascii --- )
    ASCII->ROM  CURSOR-ADDR             ( Get ROM and cursor addr )
    8 0 DO                              ( For all 8 char lines )
      OVER I + C@                       ( Get a display byte )
      INVERSE @ IF                      ( If INVERSE flag on )
        255 XOR                         ( Invert screen data )
      ENDIF
      TEXT/LABEL @ IF                   ( If TEXT mode )
        OVER I 40 * + C@                ( Get the current video line )
        255 XOR OR                      ( Remove cursor, add ROM line )
      ENDIF
      OVER I 40 * + C!                  ( Move text char to screen )
    LOOP
    DROP DROP                           ( Drop ROM and screen addr )
    CURSOR  C-RIGHT ;       -->         ( Move to next video position )
```

```
SCR # 52
( CASE STATEMENT BY DR. C. E. EAKER )
( FROM FORTH DIMENSION, VOL II, NO. 3, PP. 37-40 )

: DOCASE  ?COMP CSP @ !CSP 4 ; IMMEDIATE
: CASE 4 ?PAIRS COMPILE OVER COMPILE = COMPILE OBRANCH
  HERE 0 , COMPILE DROP 5 ; IMMEDIATE
: ENDCASE 5 ?PAIRS COMPILE BRANCH HERE 0 ,
  SWAP 2 [COMPILE] ENDIF 4 ; IMMEDIATE
: ENDCASES 4 ?PAIRS COMPILE DROP
  BEGIN SP@ CSP @ = 0= WHILE
    2 [COMPILE] ENDIF REPEAT
  CSP ! ; IMMEDIATE

0 VARIABLE C-FLAG                     ( Stop flag for TEXT/LABEL )

-->
```

```
SCR # 53
( OVERLAY TEXT UNTIL <ESC> )
: TEXT                                ( --- )
  0 C-FLAG !   1 TEXT/LABEL !         ( Init to no quit, set to TEXT )
  CURSOR                              ( Initialize the cursor )
  BEGIN                               ( Loop until <ESC> pressed )
    KEY DOCASE                        ( Do case upon input key )
     31 CASE    C-RIGHT  ENDCASE      ( Atari right arrow )
     30 CASE    C-LEFT   ENDCASE      ( Atari left  arrow )
     28 CASE    C-UP     ENDCASE      ( Atari up    arrow )
     29 CASE    C-DOWN   ENDCASE      ( Atari down  arrow )
     27 CASE 1 C-FLAG !  ENDCASE      ( Atari ESC key )
    155 CASE    C-RETURN ENDCASE
          DUP SHOW                    ( Else put on screen if can )
    ENDCASES
  C-FLAG @ UNTIL                      ( Do until ESC pressed )
  CURSOR ;              -->           ( Reset final cursor )
```

```
SCR # 54
( REPLACE TEXT UNTIL <ESC> )
: LABEL                               ( --- )
  0 C-FLAG !   0 TEXT/LABEL !         ( Init to no quit, set LABEL )
  CURSOR                              ( Initialize the cursor )
  BEGIN                               ( Loop until <ESC> pressed )
    KEY DOCASE                        ( Do case upon input key )
     31 CASE    C-RIGHT  ENDCASE      ( Atari right arrow )
     30 CASE    C-LEFT   ENDCASE      ( Atari left  arrow )
     28 CASE    C-UP     ENDCASE      ( Atari up    arrow )
     29 CASE    C-DOWN   ENDCASE      ( Atari down  arrow )
     27 CASE 1 C-FLAG !  ENDCASE      ( Atari ESC key )
    155 CASE    C-RETURN ENDCASE
          DUP SHOW                    ( Else put on screen if can )
    ENDCASES
  C-FLAG @ UNTIL                      ( Do until ESC pressed )
  CURSOR ;              -->           ( Reset final cursor )
```

```
SCR # 55
( PUT A BORDER AROUND THE GRAPHICS NOTEPAD )

: BORDER                              ( --- )
    1   0   0 PLOT                    ( Start in upper left corner )
    1 319   0 DRAW                    ( Draw counter clock wise )
    1 319 159 DRAW
    1   0 159 DRAW
    1   0   0 DRAW ;

;S
```

```
SCR # 56
( SAVE AND RESTORE SCREEN -- REQUIRES 6400 BYTES )

0 VARIABLE WORKSPACE-BUF   6398 ALLOT  ( Image working space )

: SAVE-SCREEN                         ( --- )
    SCREEN                            ( From the GR.8 screen mem )
    WORKSPACE-BUF                     ( To the working space )
    6400                              ( 160 rows x 40 bytes each )
    CMOVE ;                           ( Move the image to mem )

: RESTORE-SCREEN                      ( --- )
    WORKSPACE-BUF                     ( From the working space )
    SCREEN                            ( To screen memory )
    6400                              ( 160 rows x 40 bytes each )
    CMOVE ;                           ( Move the mem to image )
;S
```

MICRO

Directory Menu for the Color Computer

by Steve Kronschnabel and Phil Daley

The CoCo version of the menu program offers a choice of a directory menu or a directory backup onto the unused portion of track 17. This is helpful since TRS-80 DOS is noted for occasionally clobbering the directory sectors and essentially ruining the entire disk. With this program it is possible to back up your directory whenever you want and restore it to its former state should a sector become damaged. This program will save up to 48 directory entries, enough for most full disks: you would have to have a great number of short programs to fit more than 48 on one disk.

After it is determined whether you wish to back up the directory or run the menu program (line 30), lines 60-150 read the directory and store the names of the entries in N$( ) and the extensions in E$( ). Lines 170-250 print out the directory names two across, then the program waits for a choice to run. All BASIC programs will work with this menu; machine-language programs may or may not work depending upon their load address. Among my collection, EDTASM+ works, while many games do not.

Lines 370-400 determine whether the directory is to be saved or replaced due to disk crash. Lines 410-430 perform a fail-safe prompt, in case of a mistake, because the replace will cause a loss of any programs saved since the last backup. Lines 470-560 do the actual work. The same FOR-NEXT loop is used for saving or replacing the information.

This program, although simple in concept, works great and will save your sanity if your disk system crashes. Also, the menu program can be added to your disk to make it easy for beginners to choose a program to run.

Listing 1

```
10 CLEAR2000:CLS
20 DIMN$(72),E$(72)
30 PRINT@64,"mENU OR dIRECTORY"
40 A$=INKEY$:IFA$=""THEN40
50 IFA$="D"THEN370
60 CLS:PRINT@64,"reading directory"
70 FORX=3TO11
80 DSKI$0,17,X,A$,B$
90 IFASC(A$)=255THENX=11:GOTO150
100 C$=A$+LEFT$(B$,127)
110 FORN=0TO7
120 N$((X-3)*8+N)=MID$(C$,N*32+1,8)
130 E$((X-3)*8+N)=MID$(C$,N*32+9,3)
140 NEXTN
150 NEXTX
160 CLS:PRINT@10,"directory"
170 FORX=0TO72
180 Y=Y+1:A$=""
190 IFASC(N$(X))=255THENX=72:GOTO250
200 IFASC(N$(X))=0THENY=Y-1:GOTO250
210 GOSUB340
220 PRINTX$" "N$(X)"."E$(X)" ";
230 IFY/28=INT(Y/28)THENPRINT@480,"";:
     INPUTA$:PRINT@32,"";
240 IFLEN(A$)<>0THEN280
250 NEXT
260 PRINT@480,"";
270 INPUTA$:IFLEN(A$)=0THENY=0:GOTO160
280 A=VAL(A$)
290 IFE$(A)="BAS"THEN320
300 IFE$(A)="BIN"THEN330
310 CLS:PRINT"CAN'T RUN THAT PROGRAM":END
320 LOADN$(A),R
330 LOADM N$(A):EXEC
340 X$=STR$(X):X$=MID$(X$,2)
350 IFX<10THENX$="0"+X$
360 X$=LEFT$(X$,2):RETURN
370 CLS5:PRINT@32,"sAVING OR
      rEPLACING DIRECTORY?"
380 A$=INKEY$:IFA$=""THEN380
390 IFA$<>"S"ANDA$<>"R"THEN380
400 IFA$="S"THEN450
410 CLS7:PRINT@64,"ARE YOU SURE THAT YOU WANT TO
     REPLACE THE DIRECTORY?"
420 INPUT"yES OR nO? ";A$
430 IFA$<>"Y"THENCLS:END
440 X1=10:X2=0:GOTO460
450 X1=0:X2=10
460 E=0
470 FORX=2TO8
480 DSKI$0,17,X+X1,A$,B$
490 DSKO$0,17,X+X2,A$,B$
500 DSKI$0,17,X+X2,C$,D$
510 DSKI$0,17,X+X1,A$,B$
520 IFC$=A$ANDD$=B$THEN560
530 E=E+1:IFE=1THEN470
540 CLS5:PRINT@64,"TOO MANY READ OR WRITE ERRORS
      CANNOT CONTINUE..."
550 END
560 NEXTX
570 CLS1:END
```

Animated Graphics Routines for the 6809

by Craig Carmichael

This versatile package of related graphics subroutines displays moving animated figures, moving dots, and custom characters.

Have you ever wanted to write a machine-language program that required moving graphics (such as an arcade-type game) and not known where to start? Faced with this problem, I developed a set of general-purpose graphics subroutines that can be used for a wide variety of programs where moving graphics are required.

These graphics routines operate in the Color Computer's G6R or G6C graphics modes but are easily adapted for other display modes or other computers. (I first developed them on a homemade computer.)

Animated Graphics
requires:
6809 computer with memory-mapped graphics display such as CoCo.

The animated figures generated by these routines are eight bits (one byte) wide by "n" bits tall, as selected by the user. In mode G6R the byte will be divided into eight dots, while in G6C it will be divided into four double-width dots, each of which can assume four colors.

All values necessary for printing the figures are stored at positive offsets from the "U" register — from 0,U to $D,U. Other information pertinent to the figure (e.g., its status, fuel, or score) may be stored at other offsets near the U register. To select different figures, simply move the U register.

In addition to animated figures, there are routines for moving a single bit (for bullets) and for printing in specific columns to display numbers, letters, and custom characters while in graphics modes (score, fixed scenery, text).

Speed is essential to moving graphics programs; a move of one 8 × 8 character requires replacement of 32 somewhat disordered bytes of video RAM with new data — in a short time. I move all my figures during the vertical blank period of the TV scan, which is indicated by the 6847's FS* signal. That way, the figures are displayed clearly (without flickering or interference) once in each position through which they move. The available time is just over 4 milliseconds. With a 3.579MHz CPU clock (Color Computer normal speed), these routines will move in this time: two 8 × 8 figures plus several bullets, three 8 × 7 figures, or one figure up to 8 × 26.

If more figures are to be moved, I double their velocities and move them on alternate scans. The motion may not appear smooth if displays are updated less often, but one move per three or four scans may be acceptable if the velocity is low. There is no limit to the velocity a figure may have except its visual appearance on the screen.

If a figure passes through scenery or other figures, it will change color as it crosses, but afterwards both the figure(s) and the scenery will be unchanged.

If any print extends off the right or bottom of the screen, the other end of the figure will be printed on the left or top of the screen. If a figure or bullet moves off any edge of the screen, it will reappear on the opposite edge.

Descriptions of the Routines

PRGR and PRGX

These animated character print/erase routines will print or erase a figure at any screen coordinate. The two routines are identical except that PRGR checks for collisions and PRGX doesn't; consequently PRGX runs faster. Both routines affect the X, Y, and D registers.

The secret to the print/erase feature is the use of exclusive-or between the figure and the current TV picture. The "1" bits in the shape table will remain 1's if they are printed onto a blank (0) bit on the screen, but if they are printed onto a "1" bit, they will become a 0. Thus, the second time the character is printed into the same place, the screen is restored to its initial value.

The collision checks also make use of this feature. If any "1" bits of the shape table become 0's on the screen, a collision is registered, and "Z" in the CC register will be cleared at RTS, allowing tests by BEQ (no collision) and BNE (collision) instructions.

It should be mentioned here that in four-color mode, a color with a bit pattern "01" (yellow or cyan) will never register a collision with a "10" bit pattern color (blue or magenta), since the "1" of one will always land on the "0" of the other.

Notice that a collision will always be registered when erasing because the figure has "hit" itself.

PRGX and PRGR actually print one bit position to the right of the specified position; you must allow for this when checking the specific screen position of a figure.

MOVG and MOVC

These routines move figures around the screen and change their animations after they have been printed once by PRGX or PRGR. MOVG is for two-color modes (G6R) and MOVC is for four-color modes (G6C). Their operation sequence is:

1. Erase present figure with PRGX.
2. Add (two's complement) VVEL to VPOS and correct for off-screen position if necessary.
3. Add HVEL to HPOS. In the case of MOVC, HPOS units will always be set to an odd number to maintain the figure's correct shape and color.
4. Update PST to present animation.
5. Print new figure with PRGR.
6. Restore D, X, and Y, and return from subroutine.

Make your initial collision checks after a MOVe.

My method for finding *what* you've hit is to erase the figures that you could have hit, one at a time, then erase and reprint your figure. If you are no longer hitting anything, you've erased that figure with which you were in contact. If none of them check out, you must be contacting the inanimate playfield. If anyone can find a shorter method, please write!

Figure 1: Operands used by the routines, stored at positive offsets from the "U" register.

OFFSET	NAME		MOVG MOVC	PRGR PRGX	MOVB	PRGB	PRPX PRPXD
0,U	VPOS	-units	X	X	X	X	X
1,U		-fraction	X		X		
2,U	HPOS	-units	X	X	X	X	X
3,U		-fraction	X		X		
4,U	VVEL	-units	X		X		
5,U		-fraction	X		X		
6,U	HVEL	-units	X		X		
7,U		-fraction	X		X		
8,U	HEIGHT		X	X			X
9,U	ANIMATION		X	X			X
$A,U	ZERO SHAPE TABLE		X				X
$B,U	(ZST)		X				X
$C,U	PRESENT SHAPE TABLE		X	X			
$D,U	(PST)		X	X			

(Operands used by each routine: MOVG/MOVC, PRGR/PRGX, MOVB, PRGB, PRPX/PRPXD)

PRGB

This routine prints or erases a single bit on the screen instead of a graphics character, but otherwise it is the same in operation as PRGR. In addition to printing bullets, this print may be used as a building block where a print is built up one dot at a time. D and X registers are affected.

MOVB

This routine is similar to MOVG, except that it moves a single bit (the bullet) instead of an animated figure. As with MOVG or MOVC, don't forget to do an initial PRGB to print the bit.

PRPX and PRPXD

These prints ignore the three least significant bits of HPOS and thus print on byte boundaries in 32 distinct columns. They are also absolute prints, unlike the PRG s; the original value of the screen is replaced by the print's shape table. These prints always use the figure's current animation, unlike PRGR and PRGX, which can only update their animations with a MOVe.

PRPX prints a normal 8 × "n" bit figure, while PRPXD prints a double wide 16 × "n" bit figure, with two bytes of shape table required for each line of height.

SETT

This subroutine is called by the other subroutines. It changes the coordinates into a screen memory location, and it forms a "shift multiplier," which moves bits from one to eight bit positions to the right, depending on their HPOS.

Descriptions of the Operands Used

VPOS (0,U-1,U) is the vertical position of the top of the figure. 0,U is the actual vertical position, from 00 at the top of the screen to $BF at the bottom. The MOVe routines apply a correction factor to VPOS if the value exceeds $BF.

1,U is the fractional component, the amount left to move before the figure moves up or down a line. It is used in conjunction with the fractional part of VVEL in MOVes to effect smooth motion at any given velocity.

HPOS (2,U-3,U) is the horizontal position of the left side of the character from $00 at the left to $FF at the right. It is similar to VPOS, but without a correction factor.

(Continued on next page)

VVEL (4,U-5,U) is the two's complement vertical velocity, with integer and fractional components.

A value of $0100 (or $FF00) will move a character down (or up) exactly one bit position with each MOVe that is called. Smaller values will not move the figure every MOVe, while larger values will skip over some positions as the figure moves.

HVEL (6,U-7,U) is the two's complement horizontal velocity, similar to VVEL.

HEIGHT (8,U) is the height of figures at their tallest animation. For each line of height, a figure requires one byte of shape table.

ANIMATION (9,U) is the number that determines the shape table that will be used at a figure's next MOVe, PRPX, or PRPXD. By changing this number, a figure may be shown snapping its jaws, with or without a sword, or facing different directions, etc.

ZERO SHAPE TABLE (ZST, $A,U-$B,U) is the assigned pointer to the start of the figure's shape table(s). Shape tables must be consecutive and must all be the length, in bytes, of the HEIGHT number.

PRESENT SHAPE TABLE (PST, $C,U-$D,U) is the pointer to the shape table selected by the animation number. It is set by the MOVes and PRPXes by the following equation:

PST = ZST + (HEIGHT * ANIMATION)

Thus, if a figure is five bits high and animation two is selected, then PST will point $A bytes past ZST, allowing five bytes for animation zero and five bytes for animation one.

Variations to Suit...

Different Display Memory Address

Line 3670 of the program listing adds the upper byte of the starting address of the video display to the address obtained from the coordinates. The lower byte is assumed to be 0. Example: for display at $6800-$7FFF, use ADDA#$68. Also, lines 2050, 2280, 3010, 3280, and 3900 must be CMPX (last byte of display memory), in this case, CMPX $7FFF.

Shorter Height

G6R and G6C both map $C0 (192) lines onto the screen. If your display mode maps fewer, or if you don't want to use the whole screen, then change lines 2490 and 4090 to CMPA# (number of lines), and adjust the correction factors in the lines following these. Lines 2050, 2280, 3010, 3280, and 3900 must be CMPX# (last byte of display), and lines 2090, 2300, 3050, 3320, and 3920 are LEAX (−bytes of display memory), X.

Shorter Width

G6C and G6R both map $20 (32) bytes per line. If you want $10 bytes per line, the top of the SETT subroutine should be changed to LSLB, LSRA, RORB, LSRA, RORB, LSRA, RORB, LSRA, RORB. To properly convert the coordinates to a memory location. Lines 2455, 3735, and 4175 should include an ANDA#7F before STD 2,U for HPOS correction and, finally, all LEAX 20,X's must be changed to LEAX 10,X.

If your display maps other than such nice even numbers of bytes per line, you'll have to use the general formula in SETT to convert the coordinates to a memory location: Location = (HPOS/8) + (VPOS * (bytes per line) + start of TV memory. HPOS correction factors will have to be set up similar to the VPOS ones. Remember, these are the units of HPOS and VPOS — not the fractions.

Final Words

If you're not quite clear how it all fits yet, study the demonstration in the listing. It contains examples of all of the routines used in two-color mode. To try the four-color mode, change line 330 to LDA #$E5 to set mode G6C, and change line 1400 to LBSR MOVC. The shapes won't look right since the tables were designed for G6R, but you'll see the idea.

The "SYNC" instruction at line 1300 is the wait for vertical blank. I've left the BASIC handling the interrupt; all it does is clear the interrupt and return.

Once you have the program typed in be sure to save it, since running it will probably wipe out your source text. When you have it all working, delete the demo program, ORG the subroutines at some convenient spot, and save the machine code on tape. Be sure to write down the addresses of the labels that start the routines. You can write your programs and use equates for the subroutine addresses and simply load in the subroutine package when you are ready to test.

Now you're ready to write useful graphics programs without spending countless hours coaxing the bits to move themselves around on the screen as I have done. Good luck!

You may contact Mr. Carmichael at 820 Dunsmuir Road, Victoria, British Columbia, Canada V9A 5B7.

Listing 1: Animated Graphics Routines and Demo

```
          *
                         *ANIMATED GRAPHICS ROUTINES FOR THE 6809
                         *
                         *CRAIG CARMICHAEL.
                         *
                         *WITH DEMONSTRATION PROGRAM THAT CHECKS MOST OF THE
                         *ROUTINES
                         *
                         *
3000                     00190        ORG     $3000
                         *DEMONSTRATION PROGRAM FOR COLOR COMPUTER.
                         *PRINTS A 2 BYTE WIDE SHAPE, A SINGLE WIDE SHAPE, AND
                         *MOVES A BIRD AROUND WITH A JOYSTICK-BIRD FIRES UPWARD
                         *
                         *FIRST. SET UP GRAPHICS SCREEN TO G6R AT 0600-1DFF.
3000 10CE 0400           00250 DEMO1  LDS     #$400   *REMOVE STACK FROM SCREEN
3004 8E   FFD4           00260        LDX     #$FFD4  *SET "SAM" CONTROLLER
3007 A7   83             00270 DEM2   STA     ,--X
3009 8C   FFC0           00280        CMPX    #$FFC0
300C 22   F9             00290        BHI     DEM2
300E A7   03             00300        STA     3,X
3010 A7   05             00310        STA     5,X
3012 A7   07             00320        STA     $07,X
3014 A7   09             00330        STA     $09,X
3016 86   F5             00340        LDA     #$F5    *AND VDG
3018 B7   FF22           00350        STA     >$FF22
                         *ENABLE V BLANK INTERRUPT ONLY TO CPU.
301B 8E   FF21           00370        LDX     #$FF21
301E 8D   09             00380        BSR     CLRINT
3020 8E   FF01           00390        LDX     #$FF01
3023 8D   04             00400        BSR     CLRINT
3025 6C   02             00410        INC     2,X
3027 20   0D             00420        BRA     CLRS
3029 A6   84             00430 CLRINT LDA     ,X      *DISABLES 2 INT'S.
302B 84   FC             00440        ANDA    #$FC
```

(Continued on page 80)

Listing 1 *(continued)*

```
302D A7   84                STA    ,X
302F A6   02                LDA    2,X
3031 84   FC                ANDA   #$FC
3033 A7   02                STA    2,X
3035 39                     RTS
                    *CLEAR TV SCREEN.
3036 8E   0600      CLRS    LDX    #$0600
3039 6F   80        MOREBY  CLR    ,X+
303B 8C   1DFF              CMPX   #$1DFF
303E 23   F9                BLS    MOREBY
                    *PRINT DOUBLE WIDTH FIGURE ON BYTE BOUNDARIES.
3040 CE   2700      DRAW    LDU    #$2700
3043 CC   1010              LDD    #$1010
3046 A7   C4                STA    ,U      *VPOS
3048 E7   42                STB    2,U     *HPOS
304A CC   1000              LDD    #$1000
304D ED   48                STD    8,U     *HEIGHT AND ANIMATION
304F 30   8D 00AD           LEAX   DUBLPX,PCR
3053 AF   4A                STX    $0A,U   *SHAPE TABLE
3055 17   00E4              LBSR   PRPXD
                    *
                    *PRINT SINGLE WIDTH FIGURE ON BYTE BOUNDARIES.
3058 CE   2710              LDU    #$2710
305B CC   4000              LDD    #$4000
305E A7   C4                STA    ,U      *VPOS
3060 E7   42                STB    2,U     *HPOS
3062 CC   0800              LDD    #$0800
3065 ED   48                STD    8,U     *HEIGHT AND ANIMATION
3067 30   8D 00B5           LEAX   PIXLS,PCR
306B AF   4A                STX    $0A,U   *SHAPE TABLE
306D 17   00F8              LBSR   PRPX
                    *
                    *PRINT THE BIRD
3070 CE   2720              LDU    #$2720
3073 CC   6080              LDD    #$6080
3076 A7   C4                STA    ,U      *VPOS
3078 E7   42                STB    2,U     *HPOS
307A CC   0500              LDD    #$0500
307D ED   48                STD    8,U     *HEIGHT AND ANIMATION
307F 30   8D 00A5           LEAX   PXBIRD,PCR
3083 AF   4A                STX    $0A,U   *ZERO SHAPE TABLE
3085 AF   4C                STX    $0C,U   *PRESENT SHAPE TABLE
3087 17   0146              LBSR   PRGX
308A 6F   C8 1E             CLR    $1E,U   *NO BULLET YET
                    *MAIN LOOP: GET BIRD'S MOVES FROM JOYSTICK, AND MOVE
                    *THE BIRD AROUND. TRY SHOOTING SOMETHING.
                    *READ JOYSTICKS INTO $015A-5D
308D AD   9F A00A   LOOP    JSR    [$A00A]
                    *MAKE V.R. JOYSTK INTO VVEL.
3091 CE   2720              LDU    #$2720
3094 4F                     CLRA
3095 F6   015B              LDB    >$015B
3098 58                     LSLB
3099 58                     LSLB
309A 83   0080              SUBD   #$0080
309D ED   44                STD    4,U
                    *H.R. JOYSTK INTO HVEL.
309F 4F                     CLRA
30A0 F6   015A              LDB    >$015A
30A3 58                     LSLB
30A4 58                     LSLB
30A5 83   0080              SUBD   #$0080
30A8 ED   46                STD    6,U
                    *BULLET IN FLIGHT?
30AA CE   2730              LDU    #$2730
30AD 6D   4E                TST    $0E,U
30AF 26   20                BNE    SINC
                    *NO BULLET AT PRESENT. DOES BIRD FIRE ONE?
30B1 B6   FF00              LDA    >$FF00
30B4 85   01                BITA   #$01
                    *YES, PRINT BULLET, USING BIRD'S CO-ORDINATES.
30B6 26   19                BNE    SINC
30B8 EC   50                LDD    -$10,U
30BA 4A                     DECA
30BB ED   C4                STD    ,U
30BD EC   52                LDD    -$0E,U
30BF ED   42                STD    2,U
30C1 6F   46                CLR    6,U     *HVEL=0
30C3 6F   47                CLR    7,U
30C5 CC   FF00              LDD    #$FF00
30C8 ED   44                STD    4,U     *VVEL=-1(UP)
30CA 17   0191              LBSR   PRGB
                    *BULLET TO FLY $40 SPACES.
30CD 86   40                LDA    #$40
30CF A7   4E                STA    $0E,U
                    *WAIT FOR NEXT VERTICAL BLANK, THEN MOVE BIRD& BULLET.
30D1 13             SINC    SYNC
30D2 CE   2720              LDU    #$2720
30D5 6A   4F                DEC    $0F,U   *FLAP BIRD'S WINGS
30D7 26   0B                BNE    NOFLAP
30D9 86   10                LDA    #$10    *EVERY 16TH SCAN
30DB A7   4F                STA    $0F,U
30DD A6   49                LDA    9,U
30DF 4C                     INCA
30E0 84   03                ANDA   #$03
30E2 A7   49                STA    9,U
30E4 17   00A7      NOFLAP  LBSR   MOVG    *MOVE BIRD
30E7 CE   2730              LDU    #$2730  *POINT AT BULLET
30EA 6D   4E                TST    $0E,U   *IS THERE ONE?
30EC 27   9F                BEQ    LOOP
30EE 6A   4E                DEC    $0E,U   *END OF RANGE?
30F0 27   09                BEQ    ENDBUL
30F2 17   017F              LBSR   MOVB    *NO, MOVE IT
30F5 27   96                BEQ    LOOP    *HIT ANYTHING?
30F7 6F   4E                CLR    $0E,U   *YES, END BULLET
30F9 20   92                BRA    LOOP
30FB 17   0160      ENDBUL  LBSR   PRGB    *ERASE BULLET
30FE 20   8D                BRA    LOOP
3100      03C0      DUBLPX  FDB    $03C0   *DOUBLE WIDE SHAPE TABLE
3102      0C30              FDB    $0C30
3104      1008              FDB    $1008
3106      2004              FDB    $2004
3108      4002              FDB    $4002
310A      4002              FDB    $4002
310C      8001              FDB    $8001
310E      8001              FDB    $8001
3110      8001              FDB    $8001
3112      8001              FDB    $8001
3114      4002              FDB    $4002
3116      4002              FDB    $4002
3118      2004              FDB    $2004
311A      1008              FDB    $1008
311C      0C30              FDB    $0C30
311E      03C0              FDB    $03C0
3120      183C      PIXLS   FDB    $183C   *SINGLE WIDE SHAPE TABLE
3122      7EFF              FDB    $7EFF
3124      FF7E              FDB    $FF7E
3126      3C18              FDB    $3C18
3128      E718      PXBIRD  FDB    $E718   *4 BIRD SHAPE TABLES
312A      1818              FDB    $1818
312C      24                FCB    $24
312D      0066              FDB    $0066
312F      9918              FDB    $9918
3131      24                FCB    $24
3132      0000              FDB    $0000
3134      7E99              FDB    $7E99
3136      18                FCB    $18
3137      0000              FDB    $0000
3139      183C              FDB    $183C
313B      DB                FCB    $DB
                    *6809 ANIMATED GRAPHICS ROUTINES
                    *
                    *  BY CRAIG CARMICHAEL
                    *
                    *PARAMETERS FOR ALL ROUTINES ARE STORED AT POSITIVE
                    *OFFSETS FROM "U". SEE TABLE IN TEXT.
                    *
                    *
                    *PRPXD PRINTS "DOUBLE" WIDTH (16 BIT) CHARACTER IN ONE
                    *OF 32 COLUMNS. THE 3 LOWEST BITS OF HPOS ARE IGNORED.
                    *ORIGIONAL CONTENTS OF THE LOCATION ARE LOST.
313C 34   36        PRPXD   PSHS   Y,X,D
313E 17   00F8              LBSR   SETT
                    *FIND CURRENT ANIMATION SHAPE TABLE, PUT IN Y.
3141 A6   49                LDA    9,U
3143 E6   48                LDB    8,U
3145 D7   10                STB    <$10    *GET HEIGHT INTO <$10
3147 58                     LSLB
3148 3D                     MUL
3149 E3   4A                ADDD   $0A,U
314B 1F   02                TFR    D,Y
                    *PRINT LOOP. ADJUST X IF IT POINTS OFF SCREEN.
314D 8C   1DFF      PPPP    CMPX   #$1DFF
3150 25   09                BLO    OKAYD
3152 26   03                BNE    PASSD1
3154 30   88 20             LEAX   $20,X   *OFF CORNER CORRECTION
3157 30   89 E800   PASSD1  LEAX   $E800,X *OFF BOTTOM CORRECTION
                    *STORE NEXT BYTE OF SHAPE TABLE ON NEXT LINE OF PIXEL
315B EC   A1        OKAYD   LDD    ,Y++
315D ED   84                STD    ,X
315F 30   88 20             LEAX   $20,X   *DOWN TO NEXT LINE
3162 0A   10                DEC    <$10    *MORE LINES?
```

(Continued)

Listing 1 *(continued)*

```
3164 26    E7      02170           BNE     PPPP
3166 35    B6      02180           PULS    PC,Y,X,D
                   *
                   *PRINT SINGLE WIDTH FIGURE IN ONE OF 32 COLUMNS.
                   *SAME COMMENTS APPLY AS FOR DOUBLE WIDTH.
3168 34    36      02220 PRPX      PSHS    Y,X,D
316A 17    00CC    02230           LBSR    SETT
316D A6    49      02240           LDA     9,U
316F E6    48      02250           LDB     8,U
3171 D7    10      02260           STB     <$10
3173 3D            02270           MUL
3174 E3    4A      02280           ADDD    $0A,U
3176 1F    02      02290           TFR     D,Y
3178 8C    1DFF    02300 PPPS      CMPX    #$1DFF
317B 23    04      02310           BLS     OKAYS
317D 30    89 E800 02320           LEAX    $E800,X
3181 A6    A0      02330 OKAYS     LDA     ,Y+
3183 A7    84      02340           STA     ,X
3185 30    88 20   02350           LEAX    $20,X
3188 0A    10      02360           DEC     <$10
318A 26    EC      02370           BNE     PPPS
318C 35    B6      02380           PULS    PC,Y,X,D
                   *
                   *MOVG MOVES A GRAPHICS CHARACTER BY ERASING THE
                   *CHARACTER, CALCULATING THE NEW ANIMATION AND
                   *POSITION, AND PRINTING THE NEW CHARACTER THERE.
                   *COLLISIONS ARE CHECKED. USE BEQ NOCOL, OR BNE COLISN.
318E 34    36      02440 MOVG      PSHS    Y,X,D
3190 8D    3E      02450           BSR     PRGX      *ERASE OLD CHARACTER
3192 EC    42      02460           LDD     2,U       *FIND NEW HOR POSN.
3194 E3    46      02470           ADDD    6,U
3196 ED    42      02480           STD     2,U       *STORE IT IN HPOS
3198 EC    C4      02490 SAMECG    LDD     ,U        *FIND NEW VERT POSN.
319A E3    44      02500           ADDD    4,U
319C 81    C0      02510           CMPA    #$C0      *AND MAKE SURE ITS ON SCRN
319E 25    08      02520           BLO     OKAYMG
31A0 81    E0      02530           CMPA    #$E0
31A2 24    02      02540           BHS     HEREMG
31A4 80    80      02550           SUBA    #$80
31A6 80    40      02560 HEREMG    SUBA    #$40
31A8 ED    C4      02570 OKAYMG    STD     ,U        *AND STORE IN VPOS
                   02580
                   *FIND SHAPE TABLE FOR NEW ANIMATION.
31AA E6    49      02600           LDB     9,U
31AC A6    48      02610           LDA     8,U
31AE 3D            02620           MUL
31AF E3    4A      02630           ADDD    $0A,U
31B1 ED    4C      02640           STD     $0C,U
                   *PRINT NEW CHARACTER WITH COLLISION CHECK, AND RETURN.
31B3 8D    47      02660           BSR     PRGR
31B5 35    B6      02670           PULS    PC,Y,X,D
                   *
                   *MOVC HAS THE SAME FUNCTION AS MOVG, BUT OPERATES IN 4
                   *COLOR GRAPHICS MODE G6C (GM6). COMMENTS FOR MOVG ALL
                   *APPLY HERE.
31B7 34    36      02720 MOVC      PSHS    Y,X,D
31B9 8D    15      02730           BSR     PRGX
31BB EC    42      02740           LDD     2,U
31BD E3    46      02750           ADDD    6,U
31BF ED    42      02760           STD     2,U
                   *SINCE COLOR MODES USE 2 BITS/DOT, AND PRGR/X SHIFT
                   *THE PRINT 1 SPACE TO THE RIGHT, THE HPOS UNITS MUST
                   *ALWAYS BE AN ODD NUMBER.
31C1 44            02800           LSRA
31C2 25    D4      02810           BCS     SAMECG
31C4 48            02820           LSLA
                   *IT'S EVEN. IS MOVEMENT TO LEFT OR TO RIGHT?
31C5 6D    46      02840           TST     6,U
31C7 2A    02      02850           BPL     HEREMC
                   *JUMP ANOTHER SPACE LEFT OR RIGHT.
31C9 80    02      02870           SUBA    #$02
31CB 4C            02880 HEREMC    INCA
31CC A7    42      02890           STA     2,U
                   *FROM HERE, SAME AS MOVG, SO GO THERE.
31CE 20    C8      02910           BRA     SAMECG
                   *
                   *PRGX PRINTS OR ERASES AN 8 BIT WIDE BY "H" BITS TALL
                   *GRAPHICS CHARACTER. THERE IS NO COLLISION CHECK. USED
                   *IN G6R (GM7) THE FIGURE WILL BE 8 BITS WIDE BY "H"
                   *BITS TALL. IN G6C (GM6), IT WILL BE 4 DOUBLE WIDE
                   *COLORED DOTS WIDE INSTEAD OF 8 SINGLE DOTS.
31D0 8D    67      02980 PRGX      BSR     SETT
31D2 A6    48      02990           LDA     8,U
31D4 97    10      03000           STA     <$10
31D6 10AE  4C      03010           LDY     $0C,U
                   *PRINT LOOP. MAKE SURE PRINT IS ON SCREEN.
31D9 8C    1DFF    03030 PPPPGX    CMPX    #$1DFF
```

(Continued on next page)

Listing 1 *(continued)*

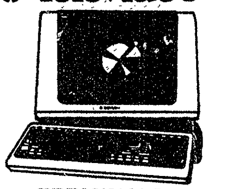
```
31DC 25   09      03040         BLO     OKAYGX
31DE 26   03      03050         BNE     PASTGX
31E0 30   88 20   03060         LEAX    $20,X
31E3 30   89 E800 03070 PASTGX  LEAX    $E800,X
                  *GET NEXT BYTE OF SHAPE TABLE, SHIFT IT RIGHT USING
                  *SHIFT MULTIPLIER AT <$11
31E7 A6   A0      03100 OKAYGX  LDA     ,Y+
31E9 27   09      03110         BEQ     ENDLNX
31EB D6   11      03120         LDB     <$11
31ED 3D           03130         MUL
                  *EXCLUSIVE OR DATA WITH SCREEN THEN PRINT IT.
31EE A8   84      03150         EORA    ,X
31F0 E8   01      03160         EORB    1,X
31F2 ED   84      03170         STD     ,X
31F4 30   88 20   03180 ENDLNX  LEAX    $20,X   *NEXT LINE DOWN ON TV
31F7 0A   10      03190         DEC     <$10
31F9 26   DE      03200         BNE     PPPPGX  *MORE LINES?
31FB 39           03210         RTS
                  *
                  *PRGR IS THE SAME AS PRGX EXCEPT IT CHECKS FOR
                  *COLLISIONS. THE SAME COMMENTS APPLY.
31FC 8D   3B      03250 PRGR    BSR     SETT
31FE A6   48      03260         LDA     8,U
3200 97   10      03270         STA     <$10
3202 10AE 4C      03280         LDY     $0C,U
3205 0F   13      03290         CLR     <$13
3207 8C   1DFF    03300 PPPPGR  CMPX    #$1DFF
320A 25   09      03310         BLO     OKAYGR
320C 26   03      03320         BNE     PASTGR
320E 30   88 20   03330         LEAX    $20,X
3211 30   89 E800 03340 PASTGR  LEAX    $E800,X
3215 A6   A0      03350 OKAYGR  LDA     ,Y+
3217 27   16      03360         BEQ     ENDLNR
3219 D6   11      03370         LDB     <$11
321B 3D           03380         MUL
                  *PRINT NEXT LINE,
321C DD   14      03400         STD     <$14
321E A8   84      03410         EORA    ,X
3220 E8   01      03420         EORB    1,X
3222 ED   84      03430         STD     ,X
                  *COLLISION CHECK ONLY THE "1" BITS OF THE SHAPE TABLE,
3224 94   14      03450         ANDA    <$14
3226 D4   15      03460         ANDB    <$15
3228 1093 14      03470         CMPD    <$14
322B 27   02      03480         BEQ     ENDLNR
                  *AND DEC <$13 IF THERE HAS BEEN A COLLISION.
322D 0A   13      03500         DEC     <$13
322F 30   88 20   03510 ENDLNR  LEAX    $20,X
3232 0A   10      03520         DEC     <$10
3234 26   D1      03530         BNE     PPPPGR
                  *SET "Z"=0/1 IF COLLISION/NO COLLISION, AND RETURN
3236 0D   13      03550         TST     <$13
3238 39           03560         RTS
                  *
                  *SET-UP TO PRINT GRAPHICS.
                  *FIRST, CHANGE CO-ORDINATES TO MEMORY LOC'N 0000-17FF.
3239 A6   C4      03600 SETT    LDA     ,U
323B E6   42      03610         LDB     2,U
323D 44           03620         LSRA
323E 56           03630         RORB
323F 44           03640         LSRA
3240 56           03650         RORB
3241 44           03660         LSRA
3242 56           03670         RORB
                  *THEN ADD $0600 TO THIS, SCREEN MEMORY IS AT 0600-1DFF
3243 8B   06      03690         ADDA    #$06
3245 1F   01      03700         TFR     D,X
                  *NOW FORM SHIFT MPYER WITH 1 BIT SET, IN"A" AND IN<$11
3247 E6   42      03720         LDB     2,U
3249 86   80      03730         LDA     #$80
324B 54           03740         LSRB
324C 24   01      03750         BCC     SHFT1
324E 44           03760         LSRA
324F 54           03770 SHFT1   LSRB
3250 24   02      03780         BCC     SHFT2
3252 44           03790         LSRA
3253 44           03800         LSRA
3254 54           03810 SHFT2   LSRB
3255 24   04      03820         BCC     SHFT4
3257 44           03830         LSRA
3258 44           03840         LSRA
3259 44           03850         LSRA
325A 44           03860         LSRA
325B 97   11      03870 SHFT4   STA     <$11
325D 39           03880         RTS
                  *
                  *PRGB PRINTS/ERASES A SINGLE BIT ON THE SCREEN.
325E 8D   D9      03910 PRGB    BSR     SETT
3260 8C   1DFF    03920         CMPX    #$1DFF  *CHECK FOR OFF SCREEN
3263 23   04      03930         BLS     OKAYGB
3265 30   89 E800 03940         LEAX    $E800,X
                  *A=SHIFT MPYER=THE DOT ITSELF. PRINT IT.
3269 97   14      03960 OKAYGB  STA     <$14
326B A8   84      03970         EORA    ,X
326D A7   84      03980         STA     ,X
326F 94   14      03990         ANDA    <$14    *CHECK FOR COLLISION
3271 91   14      04000         CMPA    <$14
3273 39           04010         RTS
                  *
                  *MOVB MOVES A GRAPHICS BIT BY ERASING THE OLD BIT,
                  *CALCULATING THE NEW LOCN, AND PRINTING A DOT THERE.
                  *IF USED IN G6R (GM7),
                  *THE DOT WILL CHANGE COLOR AS IT MOVES HORIZONTALLY.
3274 34   16      04070 MOVB    PSHS    X,D
3276 8D   E6      04080         BSR     PRGB    *ERASE OLD DOT
3278 EC   C4      04090         LDD     ,U      *FIND NEW VPOS
327A E3   44      04100         ADDD    4,U
327C 81   C0      04110         CMPA    #$C0    *MAKE SURE ITS ON SCREEN
327E 25   08      04120         BLO     OKAYMD
3280 81   E0      04130         CMPA    #$E0
3282 24   02      04140         BHS     GOBOTM
3284 80   80      04150         SUBA    #$80
3286 80   40      04160 GOBOTM  SUBA    #$40
3288 ED   C4      04170 OKAYMD  STD     ,U
328A EC   42      04180         LDD     2,U     *FIND NEW HPOS
328C E3   46      04190         ADDD    6,U
328E ED   42      04200         STD     2,U
3290 8D   CC      04210         BSR     PRGB    *PRINT NEW DOT AND RETURN.
3292 35   96      04220         PULS    PC,X,D
                  04230         END
```

MICRO


BUSIWRITER
WORD PROCESSOR PROGRAM
$99
BUSICALC 1
BUSINESS CALCULATION PROGRAM
$69
BUSICALC 2
BUSINESS CALCULATION PROGRAM
$99
WE'RE HERE!

COMPUTER MAIL ORDER

TeleVideo



TERMINALS

910 ....$559.00
912 ....$689.00
920 ....$739.00
925 ....$719.00
950 ....$929.00
970 ....CALL

COMPUTERS

800A $1099.00
802 ....$2699.00
803 ....$1949.00
802H $4695.00
806/20 ....$4999.00
816/40 ....$9199.00
1602 ....$3399.00
1603 ....CALL

MODEMS

HAYES

Smart $219.00
Smart 1200 (1200 Baud) ...$519.00
Chronograph $199.00
Micromodem 100 ....$309.00
Micromodem II $279.00
Micromodem II (with term) ...$299.00
Smart Com II $99.00
Smart 1200B $469.00

NOVATION

J-Cat $119.00
Cat $144.00
D-Cat $159.00
103 Smart Cat $189.00
Apple Cat II $279.00
103/212 Smart Cat $439.00
212 Apple Cat II $609.00
Apple Cat II 212 Upgrade $309.00

ANCHOR

Mark I (RS-232) $79.00
Mark II (Atari) $79.00
Mark III (T..I. 99) $109.00
Mark IV (CBM-PET) $125.00
Mark V (Osborne) $95.00
Mark VI (IBM-PC) $179.00
Mark VII (Auto Ans/Auto Dial) ...$119.00
Mark VIII $269.00
TRS-80 Color Computer $99.00
9 Volt Power Supply $9.00

ZENITH

ZT1 Terminal $369.00

HEWLETT PACKARD

HP 41CV...$209.00
HP 75..........$749.00
HP 41C $146.00
HP 10C $52.00
HP 11C $69.00
HP 12C $92.00
HP 15C $92.00
HP 16C $92.00

For HP41/41CV

HPIL Module $99.00
HPIL Cassette or Printer $359.00
Card Reader $144.00
Extended Functions Module ...$64.00
Time Module $64.00

EAGLE

IIE-1 ....$1369.00
IIE-2 ....$1649.00
IIE-3 ....$2399.00
IIE-4 ....$3199.00
PC-E $1579.00
PC-1 ....$2399.00
PC-2 ....$2799.00
PC-XL $3599.00
1620 ....$3599.00
1630 ....$5499.00
1640 ....$6499.00
Cyma Software ....CALL

MONITORS

AMDEK

300G $149.00
300A $159.00
310A $169.00
Color I $279.00
Color I plus $299.00
Color II $399.00
Color III $349.00
Color IV $999.00

USI

Pi 1, 9" G $99.00
Pi 2, 12" G $119.00
Pi 3, 12" A $159.00
Pi 4, 9" A $139.00
1400 Color $299.00

ZENITH

ZVM 122A $119.00
ZVM 123G $109.00

BMC

12" Green $85.00
9191 13" Color $299.00

TAXAN

12 N Green $129.00
12 A Amber $139.00

PANASONIC

TR 120 Hi-res. Green $149.00
CT 160 Dual Mode Color ...$279.00

NEC

JB 1260 ....$119.00
JB 1201 ....$149.00
JC 1212 ....$299.00
JC 12-202 ....$299.00
JC 1203 ....$469.00

GORILLA

12" Green $89.00

EPSON COMPUTERS

QX-10 ....CALL
HX-20 ....CALL

TI-40 COMPACT COMPUTER

$209

TIMEX SINCLAIR 1000 $49.95

16K Memory $44.95
2040 Printer $99.95
Vu-Calc $17.95
Mindware Printer $99.00

IBM

NEC 3550 PRINTER...$1799

PERCOM/TANDON DRIVE

5¼" 320K Floppy $279.00
5 meg hard w/controller ...$1495.00
10 meg hard w/controller ...$1795.00
15 meg hard w/controller ...$2195.00
20 meg hard w/controller ...$2595.00

AMDEK

310A Amber Monitor $169.00
DXY 100 Plotter $599.00
Color II $399.00

AST RESEARCH, INC

SIXPAKPLUS, 64K-
clock, serial, parallel $299.00
clock, serial, parallel, gameport ..$339.00
MEGAPLUAII, 64K-
clock, serial, parallel $339.00
clock, serial, parallel, gameport ..$379.00
clock, 2 serial parallel, gameport .$419.00
COMBOPLUSII, 64K-
clock, serial, parallel $299.00
I/OPLUSII-
clock, serial, parallel $145.00
clock, serial, parallel, gameport ..$179.00

PROFESSIONAL SOFTWARE

PC Plus Word Processing ...$319.00

MICRO PRO

Word Star/Mail Merge $319.00
InfoStar $299.00
Spell Star $159.00
CalcStar $159.00

MICROSTUF

Crosstalk $139.00

MICROSOFT

Multiplan $199.00

ASHTON-TATE

D-Base II $419.00

IUS

EasyWriter II $209.00
EasySpeller $129.00
EasyFiler $129.00

CONTINENTAL SOFTWARE

1st Class Mail/Form Letter ...$89.00
The Home Accountant Plus ...$109.00

SYNAPSE

File Manager $119.00

LOTUS

123 ....$369.00

PFS

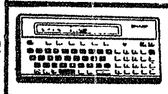
	APPLE	IBM
File	$89.00	$99.00
Report	$89.00	$89.00
Graph	$89.00	$99.00
Write	n/a	$99.00

KRAFT

IBM Joystick $55.00
IBM Paddles $39.00

QUAD RAM ....CALL

SHARP



PC-1500A $169.00
Pocket Computer

PC-1250A..........$89.00
125 Printer/Micro Cassette ...$129.00
150 Printer/Plotter/Cassette ...$172.00
152 Cassette Recorder $62.00
155 8K RAM $94.00
158 8K RAM Battery $129.00
161 16K RAM $139.00

SANYO

MBC-555PC $795.00
MBC 1100 ....$1599.00
MBC 1150 ....$2099.00
MBC 1200 ....$1999.00
MBC 1250 ....$2399.00
FDD 3200-320K Drive $399.00
FDD 6400-64K Drive $499.00
PR 5500 Printer $699.00

PRINTERS

EPSON

MX80 FT, MX100, RX80,
FX80, FX100 ....CALL

OKIDATA

82, 83, 84, 92, 93 ....CALL

STAR

STX 80 ....$169.00
Gemini 10X $299.00
Gemini 15 ....$479.00
Serial Board $75.00

SMITH CORONA

TP-1 ....$469.00
TP-2 ....CALL
Tractor Feed $119.00

C.ITOH

Gorilla $209.00
Prowriter 8510P $379.00
Prowriter 1550P $689.00
Starwriter F10-40P $1149.00
Printmaster F10-55P $1569.00
Tractor Feed $199.00

DAISYWRITER

2000 Letter Quality $1149.00
2500...NEW ....CALL
Tractor Feed $109.00

DIABLO

620 ....$949.00
630 ....$1769.00

IDS

Call for ALL Configurations on
IDS PRISM PRINTERS.

NEC

8023 ....$399.00
8025 ....$729.00
3510 ....$1449.00
3530 ....$1499.00
3550 ....$1799.00
7710/7730 ....$1999.00

BMC

401 Letter Quality $699.00

BROTHER

Comriter II Printer $499.00

CABLES & CONNECTIONS

Atari to Parallel $29.00
Atari to Serial $29.00
Apple to Parallel $69.00
Apple to Parallel/Graphics ...$99.00
Apple to Serial $89.00
IBM to Parallel $35.00
IBM to Serial $29.00
Parallel to Parallel $29.00
Serial to Serial $29.00
Grappler Plus $129.00
PKASO $139.00
Atari to Modem Cable $29.00
CBM 64 to IEEE Board $79.00
Apple 80-Column Card $159.00
CBM Pet to Parallel $99.00
CBM Pet to Serial $89.00

PAPER SUPPLIES

1 or 2" Address Labels (Tract. Feed) ..$9.95
15" Report Paper (Tract Feed) ...$24.95
8½" Blnk Wht Paper (Tract. Feed) ..$19.95

WEST

1-800-648-3311

VISA

In NV call (702) 588-5654 Order Status Number: 588-5654
Dept. 1015, P.O. Box 6689, Stateline NV 89449

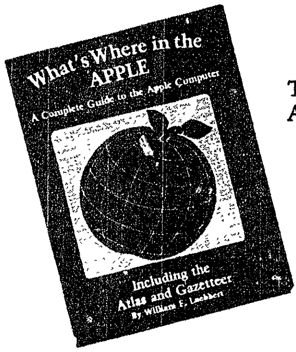
No risk, no deposit on C.O.D. orders. Pre-paid orders receive free shipping within the UPS continental United States with no waiting period for certified checks or money orders. Add 3% (minimum $5.00) shipping and handling on all C.O.D. and credit card orders. Larger shipments may require additional charges. NV and PA residents add sales tax. All items subject to availability and price change. We stock manufacturer's and third party software for most all computers on the market. Call today for our catalog.

PRODUCTS from MICRO

Revised edition now available! All new text added to the original atlas and gazetter...

"What's Where in the Apple"

A Complete Guide to the Apple Computer.



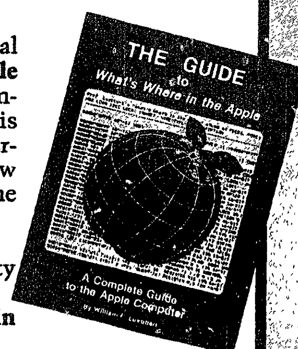
This revised edition of the famous Apple Atlas will:

- provide you with more information on the Apple's memory than is available elsewhere
- guide you with a numerical Atlas and an alphabetical Gazette — to over 2000 memory locations of PEEKs, POKEs and CALLs
- give you names and locations of various Monitor, DOS, Integer BASIC and Applesoft routines and tells you what they're used for
- enable you to move easily between BASIC and machine language
- guide you through the inner workings and hidden mechanisms of the Apple

The Atlas and the All New Guide are now available in one 256-page Wire-O-Bound book for only $24.95.

All Apple users will find this book helpful in understanding their machine and essential for mastering it.

Retail price $24.95



If you own the original **What's Where in the Apple** purchase **The Guide** to complete your edition. This 128-page Wire-O-Bound version contains all the new material to be used with the memory map and atlas.

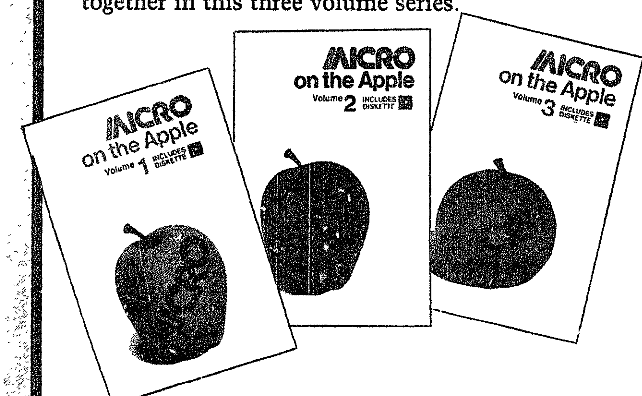
Don't miss this opportunity to complete your edition of **What's Where in the Apple.**

Order your copy of The Guide now.

Retail price $9.95

Get the most out of your Apple with the "MICRO on the APPLE" series...

This series of books for the APPLE computer is the best source of Apple specific programming aids, utilities, enhancements, games and much more. MICRO has taken the best APPLE material ever published in MICRO magazine and brought it together in this three volume series.



Volume 1 allows you to —
Round and format numbers accurately, get lower-case letters and punctuation into Applesoft strings or play the hit game "Spelunker"

Volume 2 lets you —
Speed up machine-language programming using five powerful machine-language aids, add additional editing and I/O features, or play the intriguing game "GalactiCube"

Volume 3 gives you more —
Machine-language aids, I/O enhancements, graphics and games

Each volume comes complete with diskette to save you the time of typing hundreds of lines of code.

If you use to an APPLE computer you should own these books.

Retail price $24.95

FOR EASY ORDERING
CALL TOLL FREE
1-800-345-8112
(IN PA 1-800-662-2444)
MasterCard and VISA Accepted

AND now the "MICRO on the APPLE" series is available as a boxed set.

All three voumes are packaged in an attractive box with the 110 programs on one diskette.

The set is available at a **20%** savings off the individual price.

Retail price $59.95

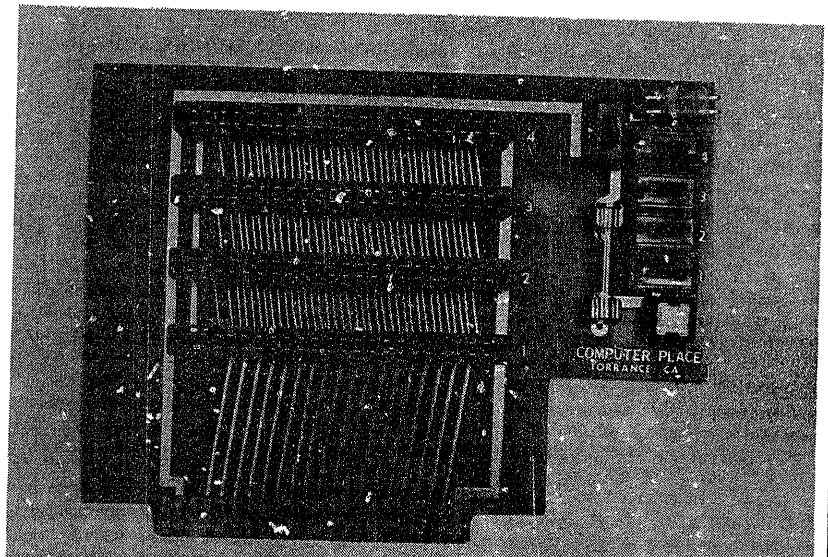
Use postage paid card opposite page 95 to order your Micro subscription today.

MICRO™

Hardware Catalog

A New Expansion Unit
for the Commodore, VIC-20 and Personal Computer

Computer Place has announced the release of a new *VIC-20* **Expander.** It differs from the others because it has such built-in features as four high-quality positively keyed connector slots for full memory expansion and utility cartridges; gold-plated contact fingers for solid, long-lasting connection; an on-board RESET button that allows the restart of the VIC-20 without turning off the computer; four individual slot ON/OFF control switches, which are arranged for easy access and designed with fingertip control rather than pentip; an external power supply hook-up provision with a two-way power source switch; and a fuse block for overload and short protection.
$54.95



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

64K of Memory
for Your Color Computer

Flexible Computer Solutions will install 64K of memory in your *Color Computer.* F.C.S. will also include a program to copy your ROM(s) to RAM and instructions on how to turn the upper 32K of RAM on and off. Extended BASIC and Disk BASIC are *not* required: *Note:* your upgraded 64K machine will behave like a standard 32K machine as far as BASIC is concerned, except that you can copy the ROM(s) to RAM and then PEEK and POKE to the RAM where BASIC does not sit; or you can actually modify BASIC if you wish. Programs that are set up to use 64K such as *Flex* and *Master Writer* will have access to the full 64K. F.C.S. will also do repairs.

For 4K, 16K, or 32K "D" or "E" boards or 16K "F" boards.
$89.00 plus $5.00 shipping. 90-day guarantee.

Flexible Computer Service
1410 W. North Loop #108
Austin, TX 78756
(512) 458-9783

Super RAM
for Atari

Mosaic Electronics, Inc., announces its second release in the RAM Select series. Named the **Mosaic 64K RAM Select,** this new board will plug into both the *Atari 400* and *800.* Atari 800 owners can plug up to three Select boards into their computer for 196,608 bytes of useable RAM. The Mosaic 64K RAM Select is totally bus-compatible for use with Atari 16K and/or Mosaic 32K RAM boards. Atari 800 owners can use the board to simulate the Atari 1200 architecture or configure the board for super powerful bank selection. The 64K Select is compatible with both 8K and 16K ROM cartridges and installs without solder. Used with the Mosaic Adapter, 48K Atari 800 owners will have 112K RAM.

64K Select owners will have access to the "Select" club that includes a monthly newsletter covering new applications for the board and a program library that includes selections such as HANDYMAN, SUPERDRIVE, and THE MEMORY MANAGER.

Mosaic Electronics, Inc.
P. O. Box 708
Oregon City, OR 97045

Color Disk Drives
for the Extended BASIC Color Computer

The **Color Disk Drive** from Radio Shack turns the Extended BASIC *Color Computer* into a disk system at a new low price. A Color Disk Drive gives 156,672 characters of user storage for program and data files. It can also be used with color-disk software.

Easy to install, the Color Disk Drive controller Program Pak is plugged into the Color Computer's cartridge port. The Color Disk Operating System is completely contained in the controller Program Pak, so the full 156K-byte disk capacity is available for on-line storage.

The first drive comes with one 5¼-inch double-density, 35-track floppy disk drive, plug-in Program Pak cartridge with cable, one blank 5¼-inch diskette, reference manual, and operator's instructions. Included cable allows up to two drives on a system.
The Color Disk #0 Kit $399.95 (26-3022)
The Color Disk #1, 2, or 3 $279.95 (26-3023) (requires Disk #0)

Tandy Corporation/Radio Shack
1800 One Tandy Center
Fort Worth, TX 76102

Hardware Catalog *(continued)*

The Apple Blooms!
for the Apple II and 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 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.
$190.00

Hollywood Hardware
6842 Valjean Avenue
Van Nuys, CA 91406
(213) 989-1204

Apple Computer Announces A New Monitor
for the Apple II Personal Computer

Apple Computer, Inc. has announced a new monochrome video display designed to blend aesthetically with its *Apple II* family of personal computers. The newly-styled **Monitor II** features superior resolution for 80-column text and graphics display, an anti-reflective, high-contrast screen, and a tilt mechanism for adjusting the screen's angle.

The monitor's 12-inch screen displays up to 24 80-character lines of text and high-resolution graphics in P31 green phosphor, a color that minimizes eyestrain. The monitor's tilt mechanism and anti-reflective, high-contrast screen also help to reduce eye fatigue in a variety of lighting situations.

The Monitor II can be used with any Apple II, Apple II+, or Apple IIe computer. Every Apple IIe computer comes with a video cable that allows the monitor to be easily connected to the computer's back panel. Video cables that were provided with Apple II and Apple II+ computers also work with the new monitor.

$229.00 (90-day warranty)

Apple Computer, Inc.
20525 Mariani Avenue
Cupertino, CA 95014
(408) 973-2042

Hardware Catalog *(continued)*

The Interfacer by Data-Cue
for the Apple II

The InterFacer by Data-Cue provides *Apple II* owners with an affordable solution to interfacing and communications. The InterFacer provides one fully programmable serial communications port. The port supports synchronous/asynchronous communications as well as both the RS-232 and the RS-422 electrical standards. The InterFacer also provides two complete parallel printer ports.

The InterFacer comes with software drivers for printers as well as a terminal emulator for communications with remote computers. This allows the connecting of a modem, high-speed data processing printer, and word-processing printer to the Apple II or Apple IIe all at the same time using a single board.

$95.00

Data-Cue
5696 HWY 431 South
Brownsboro, AL 35741
(205) 883-2933

The RAINBO-256 Analogue RGB Video Interface
for the Apple II+ and Apple IIe

The **RAINBO-256** is a high-resolution analogue RGB interface card designed to interface from *Apple II+*, *IIe* computers to Electrohome, Taxan, and other similarly interfaced color video monitors.

Conventional video monitors are composite in nature, meaning that the video signal is not separated into red, green, and blue signals (thus the name RGB). Further, using conventional monitors will limit you to the number of colors available at the output of the computer — in Apple's case only 16 colors.

The RAINBO-256 eliminates a number of problems inherent in the video circuits of the Apple, Franklin, or other look-a-likes. The video output generally is not 'clean,' meaning that there is substantial smearing among the colors. Further, when in the color hi-res mode, text takes on a variety of hues instead of being white like they should be. The RAINBO-256 solves all of these problems in one slot.

The RAINBO-256 is also programmable, unlike any other RGB board on the market today. Instead of being limited to the computer's color capabilities, the RAINBO-256 may be programmed for 256 individual colors by addressing 16 additional memory locations that the RAINBO-256 adds to the Apple.

As the output connector differs from Taxan and Electrohome, when ordering the RAINBO-256, specify the model you wish: RAINBO-256-E for the Electrohome or RAINBO-256-T for the Taxan.

The RAINBO-256 in either configuration retails for **$279.00** and is available from your authorized MICROTEK dealer.

$279.00

MICROTEK
4750 Viewridge Avenue
San Diego, CA 92123
(619) 569-0900

MICRO

New Weather-Sensing Package for Commodore 64 and VIC-20

Designed for use with *Commodore 64 and VIC-20 computers*, the new **HAWS** (Home Automatic Weather Station) from Vaisala combines a professional quality weather sensor with a creative software package that teaches, forecasts, and graphically displays weather. More than a toy or game, HAWS utilizes the same weather sensor used by weather services in 60 countries worldwide. In addition, HAWS represents the first personal computer application utilizing an external sensing device, allowing the user to interact and analyze input that is not contained in his computer or the software itself.

HAWS allows the user to monitor weather conditions inside or outside the home as well as allowing the user to interact with the software program to help predict and cope with changing weather conditions. HAWS even allows the user to rate his/her forecasting performance against the local weatherman's predictions.

HAWS is an excellent educational tool for teaching meteorology concepts and for learning about weather, either in the home or in the classroom. In addition, HAWS can also be used to monitor and control indoor living space, greenhouses, and office environments, etc.

Priced at **$199.95,** the package includes sensor, choice of cassette tape or floppy disk program, 15-foot cable with connector for the computer, and complete user manual. For more information including dealer inquiries, write or call Consumer Products, Vaisala, 2 Tower Office Park, Woburn, MA 01801; (617) 933-4500.

A Powerful Software Tool from Soft Path Systems

BRAINSTORMER for the *Apple II* with CP/M is a powerful software tool for generating potential solutions to complex problems. It works by building a description of a problem in terms of the themes and variations that affect its solution. The description of the problem is "probed" by BRAINSTORMER to generate ideas about potential solutions to the problem. The user refines the process by controlling the occurrence of particular themes and variations until a sufficient quantity of potential solution strategies is produced. Up to ten billion "idea probes" can be generated for any user-specified problem.

Potential applications for BRAINSTORMER include increasing flexible thinking, discovering new products, targeting new markets, and exploring organizational problems.

BRAINSTORMER is available for TRS80 I, III, and IV and for CP/M 80-column monitor machines including Apple II, Osborne I, and Kay-Pro II. All systems require MBASIC, two drives (5¼ SS or SD only), and 48K.

This friendly and helpful package, complete with User's Guide and example files, sells for **$50.00** if used on a single machine. A program with concurrent group license for 2-10 machines used by a single organization costs $100.00. It is available from Soft Path Systems, c/o Cheshire House, 105 North Adams, Eugene, OR 97402; (503) 342-3439.

Store Tape-based Machine-Language Programs on Disk

DISK LOADER for the *TRS-80 Color Computer* loads most 16K machine-language programs from tape to disk. This new program takes tape-based machine-language programs, stores them on disk, and allows them to run automatically. DISK LOADER is especially designed to load programs that interfere with normal disk operation. It saves multiple copies, allows renaming the program, and automatically gives program load and execute addresses. DISK LOADER is supplied on tape with easy-to-operate instructions and works with any 32K or 64K Color Computer disk system.

Price is **$13.95** ppd from Stuart Hawkinson, 6695 S. W. 203rd Court, Aloha, OR 97007.

Three Game-Format Programs to Strengthen Math Skills

Arith-Magic for the *Commodore 64* with 64K and one disk drive or tape recorder consists of three interactive game-format programs that strengthen basic math skills for elementary/intermediate level and above. Arith-Magic provides enrichment opportunities for abler students through experimentation with mathematical patterns.

Price is **$35.00** for one disk or tape and manual. Available from Quality Educational Designs, P. O. Box 12486, Portland, OR 97212-0486; (503) 287-8137.

(Continued on page 92)

HODGE PODGE Now Available for the Atari

DYNACOMP announces that **HODGE PODGE** is now available for the *Atari* . HODGE PODGE was originally written for the Apple and was well accepted by both parents and teachers. It was given top grades by several reviewers and has been included in many elementary school curriculums.

The Atari version of HODGE PODGE is almost an exact translation of the Apple version. All of the sound, color, and graphics features have been retained.

The Atari version requires 32K (cassette), 48K (disk) and will run on an Atari 400, 600, 800, or 1200 having sufficient memory. Recommended age range: 2-7.

Price: **$14.95** (cassette), **$18.95** (disk). For more information contact DYNACOMP, Inc., 1427 Monroe Ave., Rochester, NY 14618; (716) 442-8960.

Oh No! It's TROMPERS

TROMPERS from Avant-Garde is a new game for the *Apple, Atari,* and *Commodore.* Hundreds of practical-joking space critters are falling from the sky! Can you help Arnold Stump, the local dog catcher, snare them before they take over the entire city? Armed with only a net and your own skills, it's you and Arnold *versus* TROMPERS!

Available from your local dealer or from Avant-Garde Creations, Inc., P.O. Box 30160, Eugene, OR 97403; (503) 345-3043.

A Grade- and Attendance-Management Package

GradeCalc is a grade- and attendance-management package for the *Apple, Atari,* and *Commodore 64.* It is designed to free the teacher from many of the time-consuming tasks of record keeping. Grade filing and reporting are set up in a flexible manner to accommodate any teacher's existing gradebook format.

With GradeCalc the teacher has on file all the raw grades and assignment information. This file can then average grades using a variety of methods.

GradeCalc maintains attendance records in the same flexible manner as grade records. The teacher can recover a variety of reports based on the attendance records. These reports include cumulative totals of all attendance records and problem reports based on excessive absences of other problems.

The GradeCalc package is available on disk for Commodore 64, the 40- or 80-column CBM or PET computer, Apple II, Apple II Plus, or Apple IIe with at least 32K memory, and for the Atari 400, 800, or 1200 with 40K of memory. The price is **$29.95** for all versions except the Atari, which is **$34.95.** Contact Tamarack Software, Inc., Water St. Darby, MT 59829; (406) 821-4596.

A Color Compiler for the TRS-80C

Computerware introduces **THE COLOR BASIC COMPILER** for the

(Continued on page 94)

Software Catalog *(continued)*

Radio Shack Color Computer with 32K and at least one disk drive. If you have ever written a BASIC program only to find that it runs too slow to provide any action and you haven't had the courage to learn assembler, then the Color Compiler is the answer to your problem. The Color Compiler lets you write your program in easy BASIC and then converts it into fast machine language. After you run your compiled program, you may find it necessary to add some delays because the Color Compiler will make your program run an average of 40 times faster.

The Color Compiler features a total of 46 commands and functions. Most of these are a subset of Extended Color BASIC. Except for a few restrictions, you can program in BASIC and assume that anything is legal and the Color Compiler will understand. The Color Compiler generates position-independent code so that you may put the compiled program anywhere in memory, including into a ROM-pack! Don't be misled by the low price — this is a very powerful development package.

Price is **$39.95** (plus $2 shipping) and is available from Computerware dealers or directly from Computerware at Box 668, 4403 Manchester Ave., Suite 103, Encinitas, CA 92024; (619) 436-3512.

Two Sports Strategy Games

Parr Programming introduces two sports strategy games on cassette for the *Commodore VIC-20.* In **Baseball Adversary,** you make tough managerial decisions against your computer opponent. Additional players on the bench allow for pinch hitting and relief pitching.

Football Adversary places you in the role of head coach opposing the computer. Features include a selection of 31 plays and a display of half-time game statistics.

Baseball Adversary requires no memory expansion and retails for **$10.95.** Football Adversary requires 8K memory expansion and retails for **$14.95.** Available postpaid from Parr Programming, 2664 Tyler St., Gary, IN 46407; (219) 885-0611.

ARCADIA MANIA

Perseus Programming presents a fast-action game for the *Commodore 64* entitled **ARCADIA MANIA.** Nineteen different waves of creatures await your attack.

For more information or to order at **$24.95** each, write to Perseus Programming, 9311 Avery Rd., Broadview Heights, OH 4414-.

MICRO™

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)

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)

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™

P.O. Box 6502
Amherst, NH 03031

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™

P.O. Box 6502
Amherst, NH 03031

MICRO™
Interface Clinic

Interface Clinic

by Ralph Tenny

I have discussed the serial port on your computer and demonstrated experiments for the Color Computer. As I mentioned previously, the serial port on several computers is a software toggle of a PIA bit. This type of port is often referred to as a *bit banger*, in case you have heard that term. Most previous experiments depended upon the serial port being software driven, but this month I have desiged an I/O port to work with any kind of RS-232 serial port. It will give you eight bits parallel output and eight bits parallel input, driven by the serial port of your computer.

The device that allows this expansion is called a Universal Asynchronous Receiver/Transmitter (UART). UARTs are common in computers and, when used as originally intended, allow the microprocessor to write each character out on its data bus and go on about its business. The UART accepts this data, sets a "busy" flag, and serially transmits the character at a present baud rate. Simultaneously, the UART can monitor a serial input channel (at the same or a different baud rate) for incoming data. If a character is received, the UART sets a "data ready" flag to signal that it has data. Remember, computers with bit-banger serial ports must be involved with sending and receiving serial data — counting down the timing loops to match the timing of the selected baud rate. Therefore, the computer can't do anything else during serial transmission. The UART allows the processor to do something else for relatively long periods of time. This project will show you a slightly different use for a UART, but the UART will function about the same.

Let's look at a typical UART (figure 1). There are eight parallel input lines, eight parallel output lines, a serial in line and a serial out line, a clock line for each channel, and various handshake and status lines for each channel. The principle of operation for a UART is basically simple and involves having the clock speed 16 times the bit rate or baud rate. It is also important to remember that *asynchronous* transmission of serial data has another important feature — the start bit/stop bit protocol. That is, when data is not being sent, the serial line is held at a logic 1 level; when a character is sent, the serial line is taken to logic 0 for one bit time. Immediately after that, the bits are sent, one at a time, until the required number of data bits (usually eight) are sent. Then either one or two stop bits will be sent; that is, the serial line will be held at logic 1 for either one or two bit times.

It is easy to understand how the UART manages the proper bit timing to transmit a character, but how about the receive operation? Here is where the 16X-bit clock comes in. Inside the UART, a very simple circuit spins in a loop, waiting for the start bit to happen. As soon as the serial line goes to logic 0, the input circuit counts off the next eight bit clocks and samples the serial line again. If it is still at logic 0 with half a bit time used up, a start bit is recognized. Next follows a delay of 16 clock periods and the line is sampled again. That means that the logic level of the serial line is checked at about the middle of each bit time, and the 0 or 1 levels are reconstructed into an 8-bit binary word. While the stop bits are active, the input circuit is getting ready for the next character.

The description of how the various hand-shake and status lines operate will be left for you to work out. The data sheet furnished with the UART specified will aid in this research. Ignore those pins that require no connection; this will simplify the schematic diagram of the project. Table 1 shows the various pin names you need to hook up and how to strap them to obtain the performance you need.

Figure 2 shows the schematic of the serial/parallel converter. U1a and U1b plus a special crystal make a precision oscillator to drive the UART clock. U2 and U3 form a divide-by-47 circuit to change the 455 kHz oscillator to the 9600 Hz clock needed for a 600-baud interface (600 × 16 = 9600). Look at the divider network for a moment: 47 decimal is 2F hexadecimal, and U2 is a 7-bit binary counter. A full count on U2 is 7F, so if you skip the high-order output of U2 (/128) and connect the /64, /16, /8, /4, and /2 outputs to an AND gate (U3), then the output of the AND gate will go high every 47 input clock cycles. Pin 2 of U2 is the RESET pin, so U2 will divide by 47, giving the required 9600 Hz (within .8%).

The /64 output is sent to the UART for both clock inputs. U4c is a power-up RESET circuit, which performs the required initialization of the UART, while U4a and U4b generate a slow clock signal, which causes the UART to periodically sample the parallel input lines and send a serial data stream to the computer. There are two ways you can make the computer read this input: you can put the computer into a loop polling the serial input line until it gets an 8-bit character, or you can have it respond to an interrupt. Those with other computers may have to rely on polling, but the Color Computer has an interrupt input on the serial port. U4b and Q2 "tickle" the CD input of the serial port, and you can either poll the port or set up an interrupt. U4a and

(Continued on next page)

Table 1

1. Vcc - +V
2. N/C
3. GND
4. Received Data Enable - GND
5. 12 Received Data Bits
17. Receiver Clock
18. Reset Data Avail. - +V
20. Serial Input
21. External Reset
23. Data Strobe
40. Transmitter Clock
39. Odd/Even Parity Select - GND
38. # Bits/Char. 2 - +V
37. # Bits/Char. 1 - +V
36. # Stop Bits - GND
35. No Parity - +V
34. Control Strobe - +V
26-33. Data Bit Inputs
25. Serial Output

Interface Clinic *(continued)*

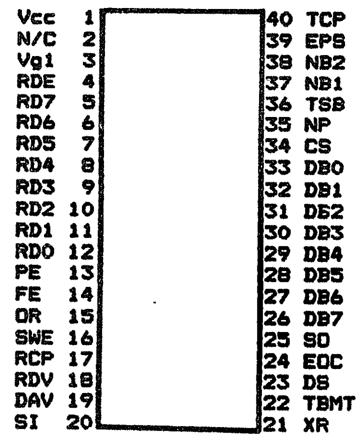
U4b generate a slow-trigger waveform, causing the UART to load whatever logic levels are on the input lines (DB7-DB0) and then transmit that binary word to the computer.

Anytime CD is pulled up by Q2, the IRQ flag in the PIA is set; if interrupts are enabled, the computer can be programmed to read the input port to service the interrupt; otherwise, the computer would have to poll the interrupt bit as was done on the previous input adapter (MICRO 63:122). U4c generates a .2-second delay as power comes on; this resets the UART. U1c and the associated resistors adapt the incoming RS-232 levels to the UART serial-in line, while Q1 is driven by the serial-out line from the UART. Although Q1 and Q2 feed the RS-232 lines on the computer, their output swings only from Vcc to ground. This will work over short distances (about 10') with no problem on most computers. If your computer fails to recognize the 0-volt signal as an RS-232 logic 1, change the 1k resistors with Q1 and Q2 to 3.9k resistors and connect them to a source of negative voltage greater than -3 volts.

When you begin to build this serial I/O adapter, you will note some simplification in the schematic and parts list, which was done to give you some practice in skills needed to design your own computer interfaces. The parts list shows some parts without Radio Shack part numbers, and no part numbers are given for resistors and capacitors. Those parts not available at Radio Shack must be obtained at one of the sources listed in earlier columns. Also, not all the pin numbers of all ICs are shown. U1, U3, and U4 are multiple gate packages, and any of the sections of the specified IC will do the required job. In addition, only the active signal lines are shown on U5, but some of the lines not shown are listed in the table, showing whether the pins must be tied to Vcc or ground. Pins of U5 not shown in either the schematic or the table are output lines that are not used; leave them unconnected. You can test the output lines (RD0-RD7) with a voltmeter or logic probe. If you type PRINT#-2 CHR$(69), bits RD6 through RD0 should contain the pattern (in binary): 1000101 or 45 hexadecimal. Input testing will be more difficult since many dialects of BASIC do not expect to receive data over the serial port. This problem will be covered next month when 2 discuss the programming.

You may contact Mr. Tenny at P. O. Box 545, Richardson, TX 75080.

***Figure 1.* Pinout for standard UARTs. Some pin name abbreviations are given in the text, and all are explained in UART data sheet.**



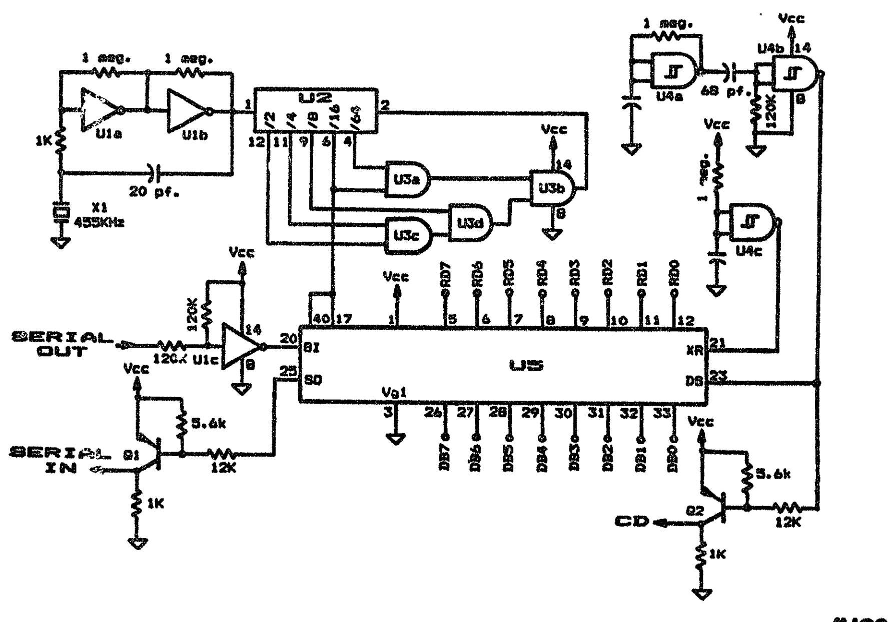


***Figure 2.* Schematic for UART-based serial-parallel adapter which converts an RS-232 serial port into a parallel port. Not all IC connections shown; see text for details.**

MICRO™ New Publications

Polishing Your Apple, Volume 2, by Herbert M. Honig. Howard W. Sams & Co. Inc., 4300 West 62nd St., Indianapolis, IN 46268, 1983, 107 pages, paperback.
0-672-22160-8 $4.95

Apple II, BASIC Programs in Minutes, by Stanley R. Frost. SYBEX, 2344 Sixth St., Berkeley, CA 94710, 1983, 176 pages, paperback.
0-89588-121-7 $9.95 plus $2.00 s/h

How To Do It on the TRS-80, Model I, II, III, Color Computer and Model 100, by William Barden, Jr. IJG Inc., 1953 West 11th St., Upland, CA 91786, 1983, 300 pages, paperback.
0-936200-08-1 $29.95

Learn to Type on Your Computer, by Frank P. Donnelly. Dictation Disc Co., 240 Madison Ave., New York, NY 10016, 1983, 33 pages, cardstock, wire-o-bound.
0-936862-16-5 $10.00

PET/CBM, An Introduction to BASIC Programming and Applications, by Gene Streitmatter. Robert J. Brady Co, a Prentice-Hall Publishing Co., Bowie, MD 20715, 1983, 341 pages, paperback.
0-89303-204-2 $13.95

The Art of Computer Programming, by Donald William Drury. Tab Books, Inc., Blue Ridge Summit, PA 17214, 1983, 303 pages, paperback.
0-8306-1455-9 $10.95

Learning Simulation Techniques on a Microcomputer, by Pat Macalugo. Tab Books Inc., Blue Ridge Summit, PA 17214, 1983, 139 pages, paperback.
0-8306-1535-0 $10.95

Computer Town; Bringing Computer Literacy to Your Comunity, by Liza Loop, Julie Anton, and Ramon Zamora. Reston Publishing Co, Inc., a Prentice-Hall Company, Reston, VA 22090, 1983, 160 pages, paperback.
0-8359-0875 $12.95

How to Build a Program, by Jack Emmerictis. Dilithium Press, P.O. Box 606, Beaverton, OR 97075, 1983, 346 pages, paperback.
0-88056-068-1 $19.95

Beginner's Guide to Reading Schematics, by Robert J. Traister. Tab Books, Inc., Blue Ridge Summit, PA 17214, 1983, 134 pages, paperback.
0-8306-1536-9 $8.95

User's Handbook to the VIC-20 Computer, by Jeffrey R. Weber and Stephen J. Szczecinski. Weber Systems, Inc., 8437 Mayfield Road, Cleveland, OH 49026, 1983, 278 pages, paperback.
0-938862-48-0 $13.95

Programming Your Atari Computer, by Mark Thompson. Tab Books Inc., Blue Ridge Summit, PA 17214, 1983, 272 pages, paperback.
0-8306-1453-2 $19.95

Computer Selection Guide, by Dan Poynter, Para Publishing, P.O. Box 4232, Santa Barbara, CA 93103, 1983, 164 pages, paperback.
0-915516-33-0 $11.95 plus $1 s/h

The DIF File, by Donald H. Beil, Reston Publishing Co., Inc., A Prentice-Hall Company, Reston, VA 22090, 1983, 235 pages, paperback.
0-8359-1305-8 $15.95

Encyclopedia of Computer Terms, by Douglas Downing, Barron's Educational Series, Inc., 113 Crossways Park Drive, Woodbury, NY 11797, 1983, 148 pages, paperback.
0-8120-2519-9 $6.95

Microprocessor Based Robotics, by Mark J. Robillard, Howard W. Sams & Co. Inc., 4300 West 62nd St., Indianapolis, IN 46268 1983, 220 pages, paperback.
0-672-22050-4 $16.95

UNIX Primer Plus, by Mitchell Waite, Donald Martin, Stephen Prata, Howard W. Sams & Co. Inc., 4300 West 62nd St., Indianapolis, IN 46268, 1983, 414 pages, paperback.
0-672-22028-8 $19.95

Your First BASIC Program, by Rodnay Zaks, SYBEX, Inc., 2344 Sixth St., Berkeley, CA 94710, 1983, 187 pages, paperback.
0-89588-092-X $9.98 plus $7 s/h

Easy Guide to Your Apple II, by Joseph Kasciner, SYBEX, Inc., 2344 Sixth St., Berkeley, CA 94710, 1983, 147 pages, paperback.
0-89588-122-5 $9.95 plus $2 s/h

MICRO

MICRO™

Reviews in Brief

Product Name: **Vanilla PILOT**
Equip. req'd: Any PET, CBM, Commodore 64, or VIC-20 (with at least 16K)
Both disk and tape versions are available
Price: $29.95
Manufacturer: Tamarack Software
Darby, MT 59829
Contact: Computer Marketing Services Inc.
300 W. Marlton Pike
Cherry Hill, NJ 08002

Description: *PILOT* is an easy-to-learn language used to teach programming and computer concepts to children and other computer novices. The Vanilla version of PILOT is an inexpensive and relatively complete implementation of the PILOT language. It has all of the standard PILOT commands such as ACCEPT, TYPE, MATCH, CALCULATE, JUMP, and USE plus additional commands for sound and turtle graphics (using the quarter-space squares in the CBM graphic-character set). The VIC and 64 versions also have special commands for color and joystick control. Any of these commands may be executed conditionally based on the status of the last MATCH. *Vanilla PILOT* allows you to enter and edit your programs using the built-in screen editor. In addition to the standard Commodore editing features, Vanilla PILOT has 19 special commands including AUTO, FIND, CHANGE, RENUMBER, and TRACE. There is even a special command to allow you to "pretty-print" your program listings on the screen or on a printer. Programs may be saved on, or loaded from, tape or disk.

Pluses: Written entirely in machine language, Vanilla PILOT is very fast. The editing commands are superb and the documentation is excellent! The turtle graphics capabilities are a great way to teach programming concepts to children and to let them have fun at the same time. *Vanilla PILOT* is a great value for the price!

Minuses: *Vanilla PILOT* lacks several important features found in other implementations of the language. The most severe limitations are: 1. calculations are limited to addition and subtraction of integers between −999 and +999; 2. no string variables are allowed, so it is not possible to use this version for the poetry- or story-writing exercises that are often a large part of other PILOTS; 3. conditional statements can only be YES or NO based on the last MATCH statement, so if you want to test for a specific numerical value of a variable, the variable must first be converted to its ASCII representation and then a MATCH made against another ASCII representation.

Documentation: *Vanilla PILOT* comes with a 113-page manual that is well-written and effectively illustrated. The first 86 pages contain a series of easy-to-follow tutorials aimed at beginning programmers. These tutorials have been classroom tested with children. The remainder of the manual is a reference guide and is aimed at the teacher and/or someone who is already familiar with PILOT.

Skill level required: No special skills required.

Reviewer: David Malmberg

Product Name: **CoCo**

Equip. req'd: Commodore 64 with either Datasette or 1541 Disk Drive

Price: $49.95

Manufacturer: ISA Software Inc. (HES)
14114 Dallas Parkway
Suite 530
Dallas, TX 75240

Description: *Coco* is an educational computer game that teaches the fundamentals of BASIC as well as the computer itself. *Via* the screen display, the program shows how memory is addressed by registers and how instructions to the computer (such as loops and subroutines) function. After taking you briefly through the keyboard's functions you are shown, step by step, how a simple BASIC program works.

Pluses: The program breaks the screen into boxes, which outline the memory registers along with the in/out functions and the commands entered. Then it shows you how everything interacts. The examples given provide a variety of programming methods and will get the novice off to a good start in understanding the BASIC language. The package has both cassette and disk versions.

Minuses: To run the program you need a Joystick Port Adapter, which I feel is easy to misplace thus rendering the program unusable. The cassette version does not have the lessons on how to use the program that the disk version has.

Documentation: The reference manual provided is clearly written and well done and allows learning the material without the computer programs.

Skill level req'd: None

Reviewer: Richard E. DeVore

Product Name:	**Word Attack**
Equip. req'd:	Apple II+, 48K, one disk drive (also IBM version)
Price:	$49.95
Manufacturer:	Davidson & Associates 6069 Groveoak Place, #12 Rancho Palos Verdes, CA 90274

Description: *Word Attack* is a four-part vocabulary building program that works. It can take the drudgery out of vocabulary drill and may be just the thing for those sagging S.A.T. scores. After you drill on three learning modes (choosing 4th- to 12th-grade words) you're ready for fun with the *Word Attack* mode. It's the old "blast the right answer" arcade game.

Pluses: The program's most powerful feature is its editor mode, which allows you to create your own word lists. I handed our 10-year-old son the documentation and a list of troublesome math terms (complete with definitions and sample sentences). He had no trouble accessing the editor and was soon demolishing the likes of "quotient," "perimeter," and "product."

Minuses: When using the editor, lines cannot be deleted except one at a time. This is a minor problem considering the overall ease of creating, editing, and saving files.

Documentation: *Word Attack* is a well-designed, well-documented program. The manufacturer has additional data disks ($19.95) for grades four through nine as well as one specifically for S.A.T. review.

Skill level required: Word lists begin at 4th-grade level and go through 12th-grade level.

Reviewer: Mario Pagnoni

Product Name:	**Bank Street Writer**
Equip. req'd:	Apple II, Apple II+, Apple IIe, Atari, and one disk drive
Price:	$69.95

Manufacturer: Broderbund Software
1938 Fourth Street
San Rafael, CA 94901

Description: *Bank Street Writer* is a word processor that is simple enough for children to use but is surprisingly sophisticated. All operations are accessible from menus. Features include page formatting, shift-key modification, file passwords, conversion of binary and text files, erase and recover, move text blocks, and find and replace.

Pluses: A complete, well-guided tutorial is provided on the master disk. Commands are continually displayed to prevent confusion.

Minuses: The 40-column, non-scrolling display makes formatting printouts difficult even with a "final draft" formatting module. Having to exit Write mode for every little correction consumes time. Text in memory is limited to about 1500 words (3200 with a 64K Apple IIe).

Documentation: Instructions are available on the disk; however, a booklet fills in any gaps.

Skill level required: Beginner

Reviewer: Mike Cherry

Product Name: **The Color Connection**
Equip. req'd: TRS-80 Color Computer 16K
Price: $29.95 tape, $39.95 disk
Manufacturer: Computerware
Box 668
Encinitas, CA 92024

Description: *The Color Connection* is a deluxe terminal package for the Color Computer. The program will work with any modem, but is specifically designed to work with the Hayes Smartmodem. All features of the Hayes modem are supported. I tested the disk version. There are two submenus accessed from the main menu: one loads the buffer and one accesses the set-up conditions.

Pluses: The program is easy to load and use. Set-up parameters include 7- or 8-bit, auto-line feed, parity, full or half duplex, phone number, macros 1-4, and save set-up file. The buffer can be transferred to disk or viewed and can be loaded from either disk or keyboard. When in the terminal mode, the buffer can be opened and closed as needed. Buffer size is over 25K with a 32K machine.

Minuses: The disk is auto executing and cannot be backed

(Continued on next page)

Reviews in Brief *(continued)*

up. Be sure to install a write-protect tab on the disk and use it only to load the program. There is no line-print utility.

Documentation: An 8-page manual is included that more than adequately explains software use.

Skill level required: No particular skill level is required.

Reviewer: John Steiner

Product Name: **Atari BASIC Compiler**
Equip. req'd: Atari 400/800, 48K, one or more disk drives
Price: $99.95
Manufacturer: Datasoft, Inc.
9421 Winnetka Avenue
Chatsworth, CA 91311

Description: The *Atari BASIC Compiler* compiles programs written in Atari BASIC into machine code. This four-pass compiler offers the option of using integer or floating-point arithmetic. An optional BASIC-to-machine code reference map may be printed to disk, screen, or printer. Assembler source files are created and saved for assembly programmers using DATASM assembler. Compiler design emphasizes speed over compactness in compiled program.

Pluses: The compiler is easy to use and fast. The choice of floating-point or integer arithmetic is important to those doing complex calculations. The line-reference map and assembler source files would be a great aid to those wishing to learn assembly language. Compiled programs are easy to load and fast running. Commercial sale of compiled programs is possible with a simple acknowledgement. The program and documentation are attractively packaged.

Minuses: The compiler does not support BYE, CONT, CLOAD, CSAVE, DOS, ENTER, LIST, LOAD, NEW, SAVE, RUN "filespec," GOTO variable, or GOSUB variable. Some minor modifications may be needed to BASIC programs before compiling.

Documentation: Concise and well organized. Sections on error handling and BASIC program optimizing are very useful.

Skill level required: Beginner/intermediate BASIC programmer.

Reviewer: Tim Kilby

MICRO

Next Month in MICRO

November features

GRAPHICS

Screen Editors — Generate your own personal character set — for the Apple, Atari, Commodore, and TRS-80 Color Computer.

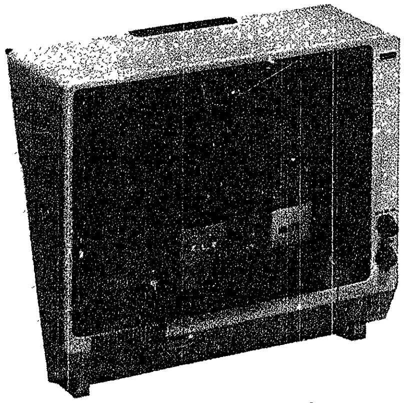
Drawing Package — An example of what you can do with your own personal character set.

Also: See our Buyer's Guide — just in time for your Christmas shopping.

Read about your computer in the new machine-specific sections — included are a bar-graph generator for the Apple and OS-9 for the TRS-80 Color Computer.

ENTER THE MICRO™ GRAPHICS

Grand Prize:



AMDEK COLOR-I MONITOR

Plus: A One Year Subscription to MICRO Magazine

4 First Prizes: AMDEK B&W Monitor
Plus: One Year Subscription to MICRO

4 Second Prizes: Box of Disks from Leading Edge
Plus: One Year Subscription to MICRO

4 Third Prizes: 1 Year Subscription to MICRO

Prizes Awarded One In Each Category: Atari, Apple, Commodore, TRS-80C

CONTEST

Use The Form Below To Enter Your Graphic Today

CONTEST RULES

1. No purchase necessary. Official Entry Forms are available in MICRO Magazine, or may be obtained, free of charge, from retail stores selling MICRO Magazine, or from MICRO Magazine directly by sending a self-addressed, stamped envelope to: MICRO Graphic Contest, P.O. Box 6502, Amherst, NH 03031. Exact copies of the Official Entry Form are also acceptable.
2. All entries must be on machine-readable format and, if possible, should be accompained by a photograph or 35 mm slide of the monitor screen taken while the graphic is being displayed. Submitted material will not be returned unless a Self-Addressed, Stamped Envelope is provided. MICRO assumes no responsibility for lost slides or damaged material.
3. All entries must be postmarked no later than December 15, 1983.
4. Contestants may enter as many times as they choose, however each graphic submitted must be an original work and must be accompanied by its own entry form. Each graphic should be titled.
5. All prizes will be awarded. In the event contest judges are deadlocked on the winner of any prize, the prize will be awarded to the contestant whose entry has the earliest postmark. Substitutions are not allowed. All decisions of the judges are final.
6. By submitting a graphic to be judged, the contestant swears and affirms that the graphic is an original work created by the contestant and assigns all rights to reproduce the graphic, for no additional consideration, to The Computerist, Inc., and its divisions: Micro Ink, and MICRO Magazine. The contestant is liable for any and all litigation, court costs, and attorneys' fees resulting from his or her submission of plagiarized or stolen graphic programs.
7. Winners will be announced in the March 1984 issue of MICRO Magazine. A list of all winners may be obtained after March 1, 1984, by mailing your request and a self-addressed stamped envelope to: MICRO Graphic Contest, P.O. Box 6502, Amherst, NH 03031.
8. Employees of The Computerist, Inc., Micro Ink, and MICRO Magazine, as well as MICRO Magazine's columnists and contributing editors are ineligible.
9. The MICRO Graphic Contest is a contest of skill, talent, and programming ability and in no way constitutes a game of chance or lottery. Void where prohibited by law.

OFFICIAL ENTRY FORM

MICRO Graphic Contest Official Entry Form

Name ______________________

Street ______________________

City ____________ State ______ Zip ____

Phone ______________________

Age ______________________

System ______________________

Graphics Package ______________________

Title of Graphic ______________________

I have read the MICRO Graphic Contest rules and understand and have followed all regulations. ______________________

Signature

Complete and mail with your graphic to: MICRO Graphic Contest
P.O. Box 6502, Amherst, NH 03031

Apple*, Commodore*, Timex/Sinclair* . . .

by William B. Sanders

The idea of getting your own computer sounded wonderful. But now that you have it you're a little scared . . . you think it sounds so technical. Well, take heart. Relax. Help is here. William B. Sanders has written individual books about the Apple, the Commodore 64, and the Timex/Sinclair computers. When you select the one which matches your computer you can breathe easy because it'll be like having your all-time favorite teacher at your side . . . gently guiding you, explaining, and showing.

THE ELEMENTARY series sweeps away the confusion and explains your Apple, Commodore 64 or Timex/Sinclair in down to earth terms, coupled with enjoyable cartoons. It shows you how to hook it up, how to use the keyboard and work on the screen—all the unique things your computer can do so you can make use of it right away! And it also answers those questions you'll have about how to write your own simple programs, about graphics, utility programs, and various hardware options.

You'll see your ELEMENTARY book contains a lot of information. And, you'll also see that not one paragraph or chapter is dull or difficult to follow. Prove it your yourself. Visit your computer store. Open the book. Read a page of the introduction, then flip to any page and read a paragraph or so. You'll find it's as understandable, as helpful and as marvelous as we say.

We make them elementary.

DATAMOST™ INC.

8943 Fullbright Ave., Chatsworth, Ca. 91311. (213) 709-1202.

VISA/MASTERCHARGE accepted. $2.00 shipping/handling charge. (California residents add 6½% sales tax)

*Apple, Commodore 64, and Timex/Sinclair are trademarks respectively of Apple Computer, Inc., Commodore Business Machines, Inc. and Timex Computer Corp.

Advertiser's Index

AB Computers 93
Acorn Software Systems 10
Alternative Energy Products 94
Amplify 20
Apple Tree Electronics 100
Arbutus Total Soft 92
Ark Computing 6
Aurora Software 100
Avalon Hill 45
Check-Mate 57
Commander Magazine 10
Communications Electronics 19
CompuTech 23
Computer Expositions, Inc. 8
Computerist 70
Computer Mail Order 84-85
Computer Marketing 1
Computer Science Engineering 14
Computer Software Association 101
D & N Micro 99
Datamost, Inc. 27, 110
Disk World 41
Eastern House Software 13
Estes Engineering 75
Excert 104
Granite Computer 106
Hollywood Hardware 82
InComm 105
Inter-Action 103
Interesting Software 88
John Bell Engineering 37
Leading Edge . . . . . Cover IV
Manx Software 23
Mercury Micro, Inc. 15
Microbits 74
MicroMotion 107
MicroSpec 15,30
Microware Distributing 20
Midwest Micro 40
Modular Mining 89
Monarch Data 63
Nibble 61
Penguin Software . . . . . Cover II
Percom Data 3
Performance Micro 63
Pterodactyl Software 106
Pion, Inc. 107
Professional Business Forms 75
Prometheus . . . . . Cover III
Protecto Enterprises 21, 51, 95
R H Electronics 43
Richvale Telecommunications 46
Scientific Software 16
SGC 17
SJB Distributors 24
Skyles Electric Works 83
Spectrum Projects 16
Star Micronics 67
Strom 41
Unique Data 81
Versa Computing 9
Victory Software 91
Winders & Geist Inc. 11
Zanim Systems 53,102

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.

• •Growth in the availability of microcomputers created a **16.1 percent increase** in the number of actively **marketed computer systems** from January 1, 1983 to July 1, 1983, according to an analysis of the DATA SOURCES data base. The analysis, a regular feature of the quarterly DATA SOURCES directory, was based on 1,893 computer systems from 566 vendors. **Nearly a third of all the systems actively marketed are microcomputers.** The high-end microcomputer vendors led the activity with a 61.1 percent increase in systems and the introduction of 44 new products. Sixteen new portable computers were announced representing a 55.2 percent expansion, and 77 desktop and personal computer units were unveiled for an increase of 23 percent. In all, this amounted to more than one new microcomputer-based system for every working day in the time period analyzed.

Such areas as mainframes, minicomputers, small business systems and board-level computers showed only a 7.5 percent growth in new products. The number of vendors marketing these systems actually declined 2.25 percent from 222 companies on January 1 to 217 companies on July 1. For more information write DATA SOURCES, 20 Brace Road, Cherry Hill, NJ 08034.

• •The **FORTH National Convention** will be held **October 14-15, 1983** at the Hyatt **Palo Alto** and will focus on FORTH-Based Systems. The convention is sponsored by the FORTH Interest Group (FIG) and is prepared to meet the needs of FORTH enthusiasts — from beginner to professional — with two days of hands-on tutorials, exhibits/vendor booths, lectures and discussions. For further information call the FIG HOT LINE (415) 962-8653 or write the FORTH Interest Group, P.O. Box 1105, San Carlos, CA 94070. Registration is $5.00.

• •A manufacturer-sponsored **credit card** in the retail computer industry has been announced by General Electric Credit Corporation (GECC) and Apple Computer. The **Apple Card** is the centerpiece of a new consumer credit financing program for buyers of Apple computers and compatible hardware and software. The program has been designed by GECC for Apple's U.S. network of 800 authorized retailers representing over 1400 stores. It will be in place for consumers to use on July 15th.

The Apple Card will enable customers to purchase Apple computers and system components without drawing on other lines of credit. Finance charges on Apple Card balances will be competitive with bank card charges. Applications for an Apple Card are available at any participating authorized Apple U.S. dealer. To qualify for the credit card, the consumer must purchase an Apple personal computer and finance a minimum of $825.00. Up to 90 percent of the initial purchase can be charged by qualifying customers.

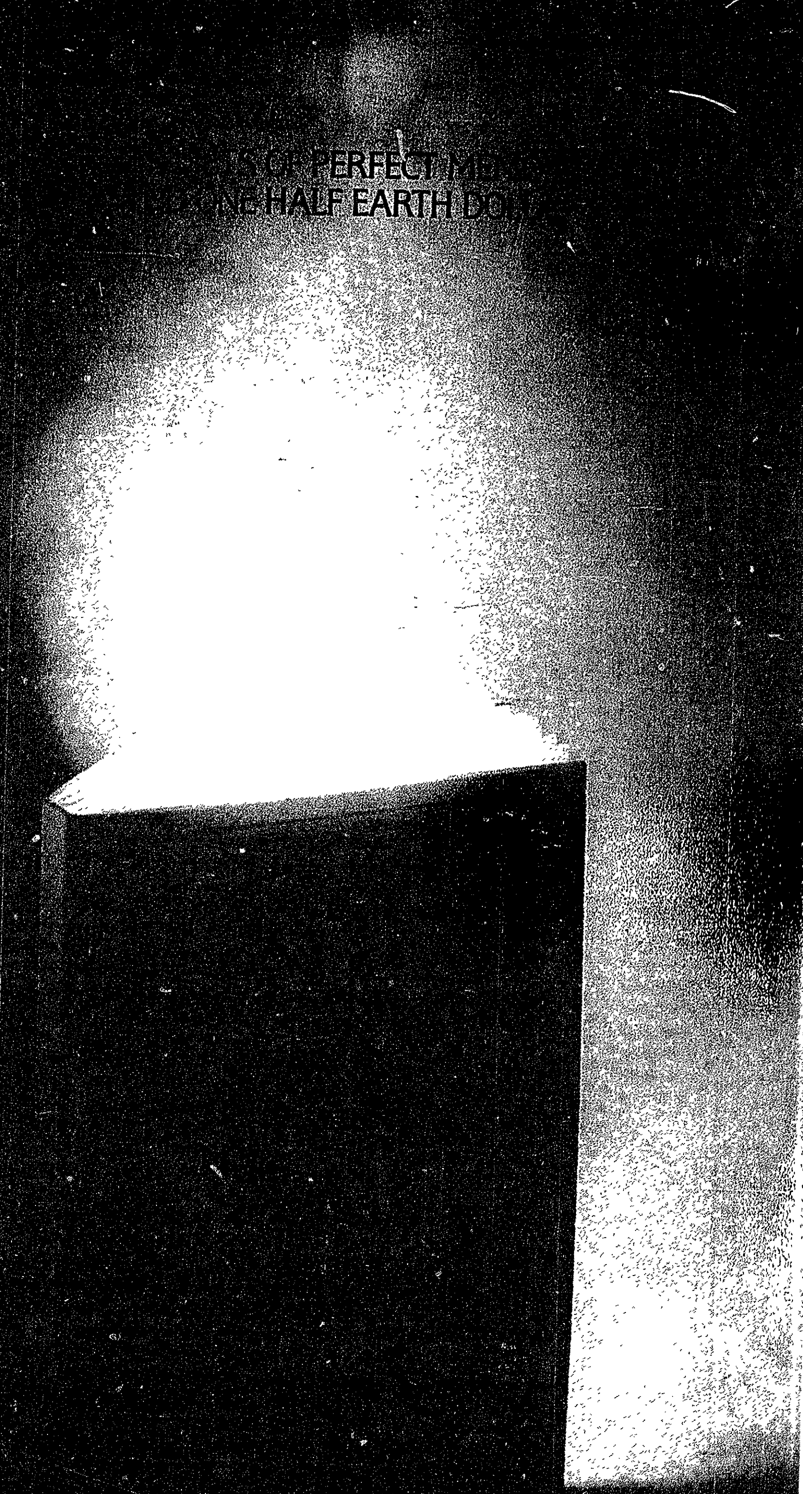
• •The **National Software Show** will be held **October 19-21, 1983** at the **San Francisco** Trade Show Center, San Francisco, CA. For more information contact The National Software Show, 21 Tamal Vista, Suite 175, Corte Madera, CA 94925.

• •"Educational Computing Profile," a **television series for educators and parents** will premiere on the **PBS network this September.** Designed to give practical guidance on buying and using microcomputer software, hardware and peripherals for educational purposes, the nine monthly half-hour programs will update parents, educators, and librarians on the latest technical developments and issues in the microcomputer field.

The program will use a magazine format and include reports and interviews in the areas on trends and news, courseware evaluation, hardware evaluation, and interviews with educators. Check your local PBS listing for time and date. For further information, contact Mary Campbell at KET- (606) 233-3000, Ken Komoski at EPIE (516) 283-4922 or Barbara Garris at EPIE (212) 678-3459.

• •**CompuServe** subscribers can access FIRSTWORLD **Travel Service** in the Home Services area of the CompuServe Information Service. This travel Club publishes special ticket prices, offers a 24-hour reservation service, a catalog of discounted tours and cruises and membership to the Very Important Traveler (VIT) Club. The VIT Club retains a personal travel profile that lists your travel preferences, such as charge card instructions and flight accommodations.

CompuServe Incorporated is a computer services company providing value-added data processing services to large corporations, financial institutions and government agencies. In addition, CompuServe offers network communications services and operates computer schools in several U.S. cities. The CompuServe Information Service, the largest commercially available videotex information service, currently serves approximately 53,000 subscribers.


HALF EARTH