Complete Programs Included

NO. 70 MATHEMATICS

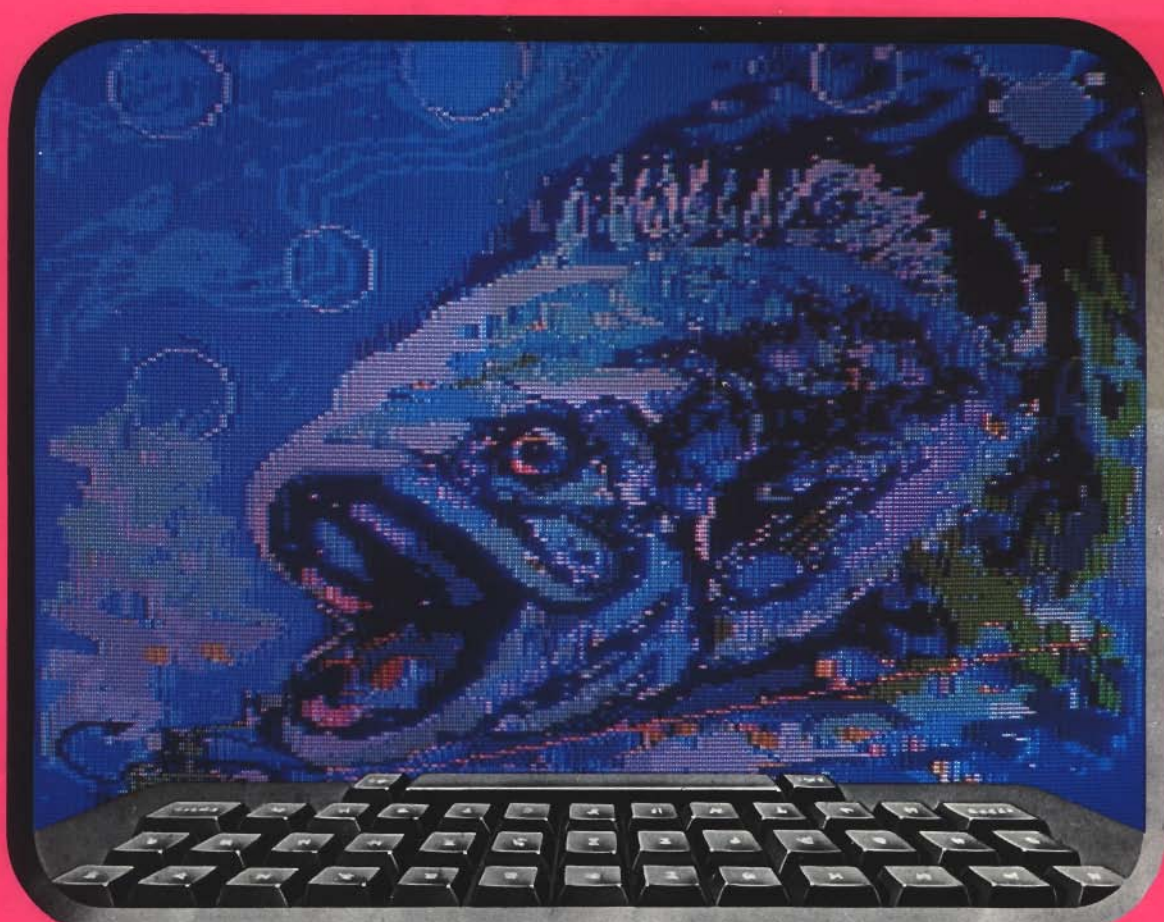
U.S. Edition: $2.50
International Edition: $3.00

MAR 84

MICRO™

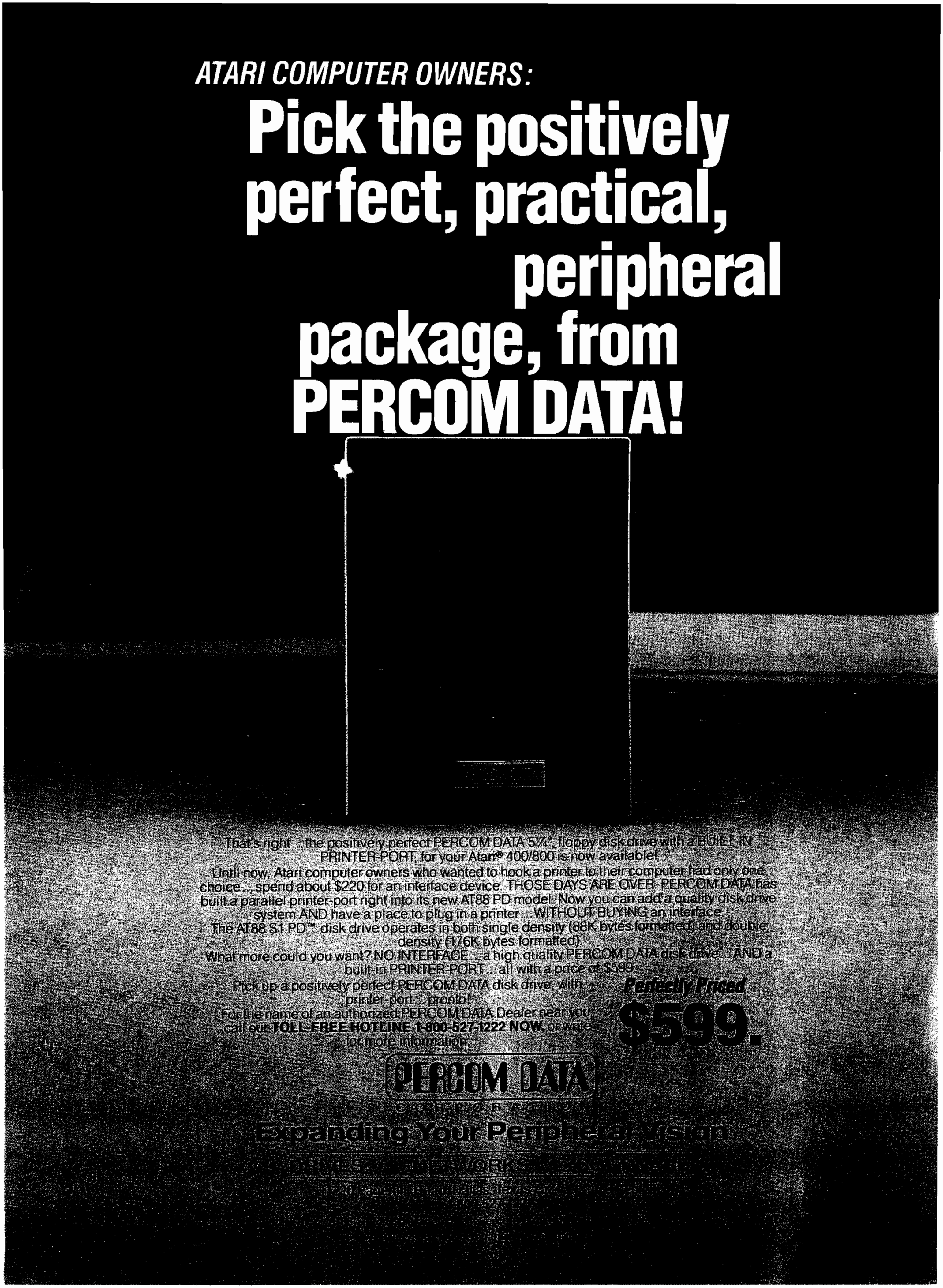
for the Serious Computerist

MATHEMATICS



* **Bezier Curves**
* **Credit Card Register**
* **Plotting Fractals**
* **Multiplication on 6809 vs. 6502**
* **Compiling BASIC Subroutines**

0 74470 11690 03


ATARI COMPUTER OWNERS:
Pick the positively perfect, practical, peripheral package, from PERCOM DATA!
That's right ... the positively perfect PERCOM DATA 5¼" floppy disk drive with a BUILT-IN PRINTER-PORT, for your Atari® 400/800 is now available!
Until now, Atari computer owners who wanted to hook a printer to their computer had only one choice ... spend about $220 for an interface device. THOSE DAYS ARE OVER. PERCOM DATA has built a parallel printer-port right into its new AT88 PD model. Now you can add a quality disk drive system AND have a place to plug in a printer ... WITHOUT BUYING an interface.
The AT88 S1 PD™ disk drive operates in both single density (88K bytes formatted) and double density (176K bytes formatted).
What more could you want? NO INTERFACE ... a high quality PERCOM DATA disk drive ... AND a built-in PRINTER-PORT ... all with a price of $599.
Pick up a positively perfect PERCOM DATA disk drive, with printer-port ... pronto!
For the name of an authorized PERCOM DATA Dealer near you, call our TOLL-FREE HOTLINE 1-800-527-1222 NOW, or write for more information.
Perfectly Priced
$599.
PERCOM DATA
CORPORATION
Expanding Your Peripheral Vision

MICRO™

Editorial

A Simple Assembly Listing ...

Having recently improved our techniques for producing BASIC listings, MICRO focused its attention this month on improving the Assembly Listing process. Producing an assembly listing may appear to be a fairly trivial task for a magazine. The author sends in a copy of his assembler's printout and you print it. This may not provide the reader with the best listings due to variations in quality, size, type of information output, and variations in listings by different assemblers. MICRO has taken a number of steps to improve the assembly listings which involved custom programming and a lot of work, but we believe the end result is worth it.

Transferring to FOCUS

The first step is to get the original listing, provided by the author in some machine readable media, onto our 6809-based FOCUS system. Techniques have been developed to transfer text between the FOCUS and the Apple, Atari, Coco and Commodore computers.

Listing Standardization

The FOCUS word processor is used to 'standardize' the listings. Its search/replace function is used to make changes quickly and accurately. The 'standard' that has been selected is the LISA 2.5 running on the Apple II. Rather than discuss the minor eccentricities of the LISA, I suggest you look at some of the listings in this issue to see the standards. A couple are worth mentioning. The LISA does not require (or accept) the **A** register designation in the **ROL, ROR, LSR,**or **ASL** instructions. It requires a special pseudo-op, **EPZ** to equate page zero addresses. These, and other minor changes, are made. Then the LISA-fied text file is transmitted to the Apple II.

Running LISA

LISA is instructed to accept input from the serial port by a **CTRL-D, IN#2, RETURN**. A transmit program on the FOCUS sends a line of text to the Apple at 1200 baud, and then waits approximately 1.5 seconds between lines to permit LISA to do its housekeeping.

LISA is instructed to assemble the file with the output directed to the FOCUS *via* serial port by the command **CTRL-D, PR#2, RETURN**.

MICRO-izing the Listing

LISA output contains more than we need for the magazine listing. A FOCUS program converts LISA output to a 'MICRO' format.The example below shows the difference in output.

Is It Worth It?

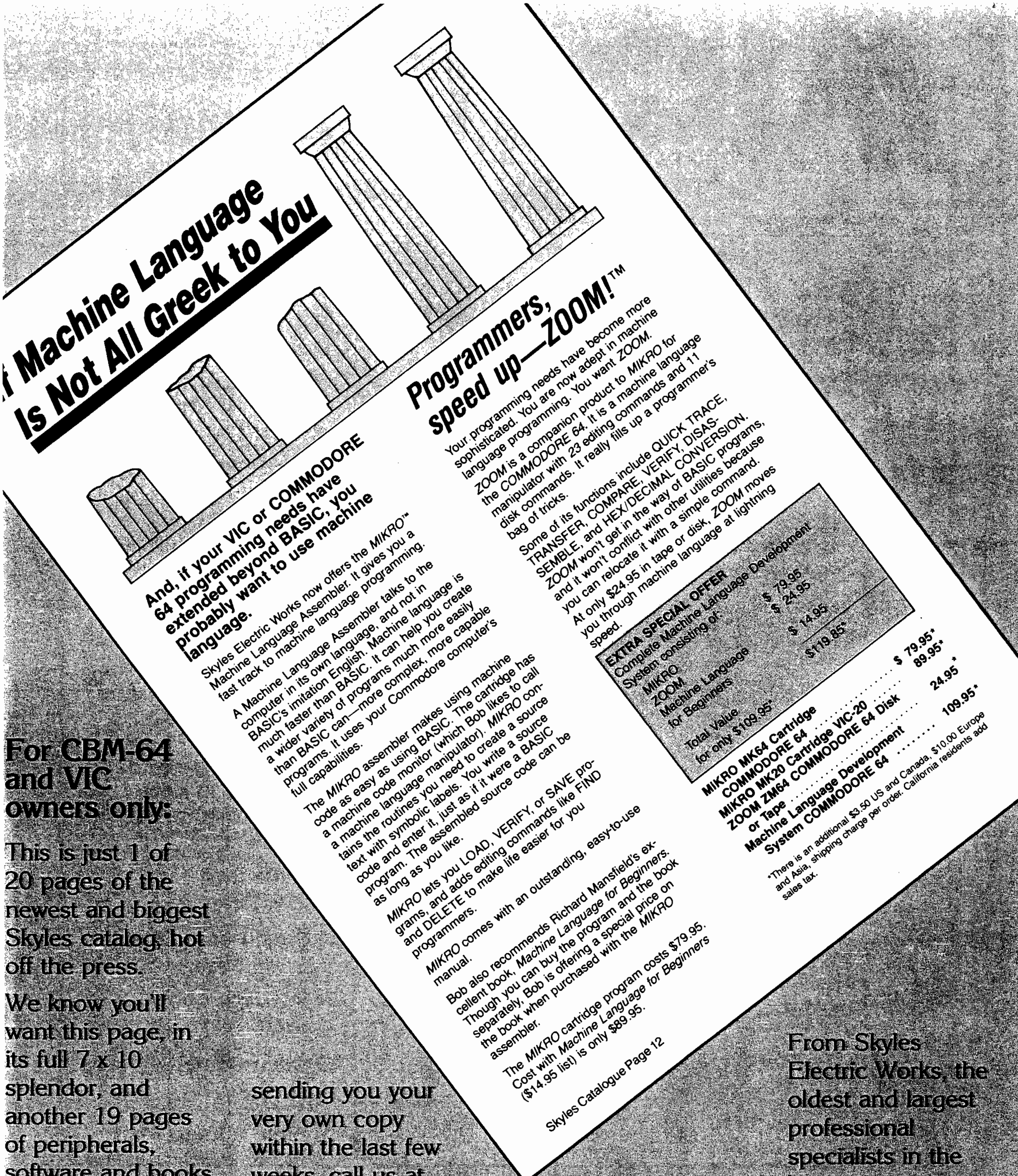
It takes a lot of work to get the listings right. Now that the special programs have been written it is easier, but still requires time and effort. Is it worth it? I think so. An important feature of MICRO is its support of assembly language programming. It is important that the listings are not only accurate, but that they are in a form that is easy for all readers to understand. We welcome, as always, your comments and suggestions.

Robert M. Tripp

Editor-in-Chief

```
0800            4  ; REQUIRES DOS+ UTILITIES @ $CXXX
0800            5  ;
C000            9         ORG $C000
C000           10         OBJ $0800
C000           11  ;
00FD           12  NUML   EPZ $FD
00FE           13  NUMH   EPZ $FE
C88F           15  D4     EQU $C88F        ;DOSPLUS ROUTINES
CFB1           21  SCRRCL EQU $CFB1
C000           31  ;
C000 20 99 CF  32  FRMPTR JSR SCRSAV       ;MAKE SURE
C003 A9 45     33         LDA #$45         ;DEFAULT IS
C041 50 52 4F  52         ASC 'PROWRITER--CONNECTION'
C044 57 52 49
C047 54 45 52
C04A 2D 2D 43
C04D 4F 4E 4E
C050 45 43 54
C053 49 4F 4E
C056 92 0D 0D  53         BYT $92,$0D,$0D
C4F7          362         END
```

```
                 ; REQUIRES DOS+ UTILITIES @ $CXXX
                 ;
C000                     ORG $C000
                 ;
00FD             NUML    EQU $FD
00FE             NUMH    EQU $FE
C88F             D4      EQU $C88F    ;DOSPLUS ROUTINES
CFB1             SCRRCL  EQU $CFB1
                 ;
C000 20 99 CF    FRMPTR  JSR SCRSAV   ;MAKE SURE
C003 A9 45               LDA #$45     ;DEFAULT IS
C041 50 52 4F            ASC 'PROWRITER--CONNECTION'
C056 92 0D 0D            BYT $92,$0D,$0D
C4F7                     END
```



For CBM-64 and VIC owners only:

This is just 1 of 20 pages of the newest and biggest Skyles catalog, hot off the press.

We know you'll want this page, in its full 7 x 10 splendor, and another 19 pages of peripherals, software and books that will make your CBM-64 or VIC computer even nicer to live with. So, if we missed sending you your very own copy within the last few weeks, call us at **(800) 227-9998** (unless you live in California, in which case call **(415) 965-1735.**

From Skyles Electric Works, the oldest and largest professional specialists in the business.

Skyles Electric Works
231-E S. Whisman Road
Mountain View, California 94041

MICRO™

for the *Serious Computerist*

MICRO
P.O. Box 6502
Chelmsford, MA 01824
617/256-3649

MICRO is published monthly by:
MICRO, Chelmsford, MA 01824.
Second Class postage paid at:
Chelmsford, MA 01824 and additional mailing offices.
USPS Publication Number: 483470.
ISSN: 0271-9002.
Send subscriptions, change of address, USPS Form 3579, requests for back issues and all othe fulfillment questions to:
MICRO
P.O.Box 6502
Chelmsford, MA 01824
or call 617/256-3649.
Subscription Rates: (per year):
U.S. $24.00 or $42.00 for two years
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, Australia, New Zealand $72.00

Features

Departments

Reviews in Brief

Product Name: **ANA-LIST**
Equip. Req'd: Apple II+ 48K, Disk Drive
Price: $150
Manufacturer: Synoptic Software, Inc.
57 Reservior Lane
Chestnut Hill, MA 02167

Description: A new list management program. It takes lists or tables of data and allows the user to rearrange the entries to his liking. The program will take advantage of the existence of a 16K card by expanding the maximum number of items in a list that can be handled.

Pluses: Without question, this program's greatest plus is its ability to use input data structured in DIF format. This, of course, makes Visicalc data entirely manageable. Few other pure data base management packages I've seen can do that. Congratulations to Synoptic for spotting an unfilled niche in the market. Another nice feature worth mentioning is the program's speed. Things happen *fast.*

Minuses: It is somewhat surprising that a package being released now by a company that has clearly done its marketing homework does not have features specifically designed to take advantage of Apple IIe capabilities. In fact, the IIe isn't enen mentioned in the manual so, if you have a IIe, you would be wise to check with the company about possible idiosyncrasies. Additionally, I thought the package was overpriced, though not so much as other similar items.

Documentation: Overall, I was pleased with the manual. The manufacturer resisted the temptation to fill the tutorial with banalities and has kept each chapter functional. I do have one complaint. For $150 the user surely deserves better paper for the manual, and he certainly deserves index tabs of plastic. The paper ones supplied will tear off within a few days of constant use. But it really is quite good.

Skill level: A novice who follows the manual should have no trouble using the product.

Reviewer: Chris Williams

Product Name: **Printmate 99 Printer**
Equip. Req'd: Serial, Parallel or IEEE Interface
Price: *
Manufacturer: Micro Peripherals, Inc.
4426 Century Drive
Salt Lake City, UT 84107

Description: A high quality dot-matrix printer with bi-directional printing, true descenders and one-pass underlining capability, the 99 prints 80characters per line. While printing normally with a 5 x 9 dot matrix, a serif font is included for letter quality type in an 11 x 9 matrix. Dot addressable graphics are supported and an Ap-pak is available for Apple owners with full software support for screen dumps including different sizes and alignment.

Pluses: A very nice, quiet, high quality printer. The graphics software is easy to use.

Minuses: The plastic cover to deaden the printing sound is inconvenient to feed the paper through.

Documentation: An 80+ page technical manual and a 50+ page Ap-pak reference manual are included. They are terse, well written manuals.

Skill level: No prior knowledge necessary.

Reviewer: Phil Daley

Product Name: **Aztec C**
Equip. Req'd: Apple II, II+, or IIe with 2 disks
Price: $199 (diskettes with software and manual in notebook
Manufacturer: Manx Software Systems
P.O. Box 55
Shrewsbury, NJ 07701

Description: Aztec C is a complete development system for writing C language programs on the Apple II. There are 3 diskettes full of goodies: *two* C compilers, a 6502 assembler, a pseudo-code assembler for assembling the pseudo-code generated by one of the two compilers, a linker, several runtime libraries, a full-screen editor, a command interpreter (called SHELL, after the UNIX command interpreter), utilities for constructing object libraries and source archives, and various other programs. The implementation of the C language seems to support the full language as specified in the book by Kernighan and Ritchie. The system uses and produces files which are in DOS 3.3 format on the disk. However, you must BRUN SHELL in order to interact most conveniently with the system.

Pluses: This is a **complete** development system. The only software that I can think of that rivals it for the price would be Apple Pascal(or the UCSD P-System). The system provides a bare bones UNIX-like environment: the SHELL provides the 5 or 6 most popular UNIX commands. You get C source code for a large part of the software provided with the system, so modifications are possible.

Minuses: You have to think hard to find any. There are some bugs in the native code generated by the C65 compiler. However, Manx is fixing them and provides updated software. Disk access performance is not optimized so there is painfully slow development time. To prepare a few line program takes almost 5 minutes (compare to UCSD's less than a minute).

Documentation: A notebook with 8 1/2 x 11 pages has 5 ample sections and two appendices. A bit terse for beginners, but adequate for seasoned programmers. There is a tutorial sections and one on technical information which provides details of system and compiler operation for more advanced users.

Skill level: Experience in using operating systems and high-level language coding and program preparation (e.g., good knowledge of Applee Pascal would be typical of level needed). Knowledge of C language is a must; buy the package and buy and read Kernighan and Ritchie.

Reviewer: Richard C. Vile, Jr.

Product Name:	**DISKEDT**
Equip. Req'd:	16K or 32K Color Computer with 1 disk
Price:	$24.95
Manufacturer:	Spectral Associates 141 Harvard Avenue Tacoma, WA 98466

Description: DISKEDT is a disk editor which is more than just a repair program. Two versions of the program are furnished on the same disk, with the 32K version having more capabilities and protection against mistakes than the 16K version. DISKEDT allows direct access, viewing and editing of any part of the disk, by track, sector or filename, which is somewhat unusual for this type of program. Disk data is displayed in either Hexadecimal or ASCII, and single-key commands allow moving forward or back through a disk or file. This moving around can be random access (specify track and sector), step-by-step or "skip to" movement. Besides the disk data, there is a constantly updated display of drive, track and sector, plus three special characters which aid in editing and generating disk data. A truly impressive set of editing capabilities are available; in fact, there are editing capabilities for which I can'timagine the purpose!

Pluses: This program works well, has very powerful capabilities and is inexpensive for all it does. Also, the display update is quite rapid so that a minimum of time is wasted.

Minuses: Although the program has considerable capability for simple disk file "repair", this topic is completely ignored. The display is highly readable, but is presented in a 10-column width. As a result, directory displays are very difficult to interpret, since the next file name occurs 32 characters "down the line".

Documentation: Well written and suitable for the disk format expert. Directions are given with the assumption that the user is intimately familiar with all facets of the

Radio Shack format and disk files in general. Although some specific examples are given, a larger number and more basic examples would have greatly enhanced the product.

Skill level: Very advanced programmers will derive much use from this product, but a dedicated computer hacker with plenty of time for study can derive experience and benefits from it.

Reviewer: Ralph Tenny

Product Name:	**STARDOS 64**
Equip. Req'd:	64K TRS-80C Color Computer with 1 disk drive
Price:	$49.95 (Disk only)
Manufacturer:	Star Kits P.O. Box 209 Mt. Kisco, NY 10549

Description: STARDOS 64 is a true Disk Operating System which will run on the Color Computer if it has 64K of read-write memory. It uses the same disk format as the Radio Shack DOS. The advantage of STARDOS is that it provides the following features: provision for multiple 320 byte File Control Blocks, routines which open, read, write and close named files, rename or delete files, read or write single sectors, search or modify the directory and other functions. STARDOS 64 will support single or double density disks, 35, 40 and 80 track disk drives, single or double sided. Finally, it has utilities which allow it to read FLEX disks and convert them to STARDOS/RS format. This means that data files and source files can be transferred from FLEX to STARDOS and then to RS Disk Basic. The standard entry points for its internal routines are the same as for the same functions in FLEX, which opens up the possibility of running some FLEX programs under STARDOS 64.

The following memory-resident commands are available: GET (load a binary file), XEQ (similar to BASIC EXEC), BAS (return to BASIC), PNS (Printer Non-Standard; adjusts for special printer protocols), VON and VOF (control disk verify), and V32, V40, V51, V64 (control special hi-res display character fonts). In addition, disk resident commands allow the user to BACKUP disks, BUILD simple text files, print a CATalog or DIRectory of disk files to screen or printer, COMPARE two disks (two drives are required), COPY files, DELETE files, DISKCHEK (test) disks for errors, LIST test files, RENAME files, SAVEM binary files and DSKINI disks. PEEK and POKE work the same as in BASIC, while ACONVERT converts FLEX ASCII files to STARDOS, BCONVERT changes binary files and FCAT reads the directory of a FLEX disk. SETMAX sets the number of tracks used on a disk, provided the drive can handle greater than 35 tracks.

Pluses: STARDOS 64 is a low cost, highly versatile DOS which is easy to use and runs on the standard 64K Color Computer. Directions are given for adding user disk commands for system expansion.

Minuses: The only minus I have noted is that STARDOS does not have a large stable of programs to support it. However, some FLEX programs will run unmodified under STARDOS and others can be converted using instructions furnished. Finally, more programs are being added as time goes on.

Documentation: An 80 page manual is furnished which explains how to use and expand STARDOS, and how to convert FLEX programs. It also gives considerable detail on proper and efficient use of a DOS. This manual is well organized, thorough and well written in a highly readable style. The manual covers both regular STARDOS (runs on unmodified 16K and 32K computers, and furnished on the same disk) and STARDOS 64.

Skill level: A DOS is essentially useless for BASIC-only operation, and almost indispensable for the assembly language programmer who does more than dabble in programming. All experience levels can benefit, but the advanced programmer will make greater use of STARDOS initially.

Reviewer: Ralph Tenny

Product Name:	**MATHMENU 1.0**
Equip. Req'd:	TRS-80 Color Computer with 16K memory (tape version) TRS-80 Color Computer with disk and 32K memory (disk version)
Price:	$44.95 tape, $49.95 disk
Manufacturer:	INTER + ACTION 113 Ward Street New Haven, CT 06519

Description: MATHMENU is a collection of 15 different engineering and math programs. The programs included will perform the following functions: Plot (both two dimensional and three dimensional); Matrix Operations (performs 8 standard matrix operations on a matrix as large as 8 x 8); Vector Operations (eight separate operations may be performed on vectors consisting of up to 20 elements each; Numerical Differentiation and Integration; Least Squares (performs least squares curve fitting); Number Base Conversions; Reverse Polish Calculation (acts as a calculator with stacks and memory visible on the screen); Binomial Expansion; Prime Number Checking; Large Add and Multiply (substitutes digits for scientific notation on large numbers); Rectangular and Polar Conversion; Quadratic Root Computation.

Pluses: Menu driven for ease of use (disk version). RPL calculator is useful and well done. Some documentation is presented on-line for each function. Algorithms used appear to run relatively fast in benchmark tests.

Minuses: Many assumptions of user knowledge level are made in the documentation. The tape version is difficult to use because programs must be separately loaded. The experienced user should have the option of skipping online documentation.

Documentation: A 24 page manual provides a general summary of functions and at least one page of detail on each program. It lacks examples and assumes a high level of math expertise, but is generally adequate in its explanations of how to use the programs.

Skill Level: These programs require a high level of math applications skills. No programming skills are required to run them.

Reviewer: Norman Garrett

Product Name:	**Homebase**
Equip. Req'd:	TRS-80 Color Computer; 32K Disk BASIC (1 drive required)
Price:	$79.95 disk
Manufacturer:	Homebase Computer Systems P.O. Box 3448 Durham, NC 27702

Description: Homebase is a complete database manager divided into three separate parts which will work alone or in unison and which may be purchased separately or as a single unit. The division has separated the package into the Data Management function, the Custom Reporting Function and the Text/Word Processing Function. A tutorial program is included which allows the new user to learn the main features of Homebase while experimenting with the pre-established database.

Database functions of formatting, adding, changing and deleting records are performed, and utilities are also included for selection, ascending or decending sorts, merges, filecopies and file synchronization. Data entry screens can be customized. Full computational functions are available, as well as a report writer that includes form letter management and full interface with the text processor and data manager, and a mailing label printing routine.

Pluses: Complete, easy-to-use tutorials are included on a separate disk and include documentation. The database is menu driven, making access rapid and efficient. The system is set up to function with Epson, Radio Shack, Okidata and NEC printers with good documentation on other models. The database manager itself is versatile and contains a number of utilities which enhance the ability to manipulate data. Another plus is the ability to backup files to and reload from cassette tape. Data entry is accomplished via user-designed data entry screens. Calculations use predefined or user-defined formulas. The report writer allows form letters using database fields and in conjunction with the text processing facility, with label printing routine built in.

Minuses: On some of the tutorial screens, words are wrapped rather than divided. The use of color and reverse video on tutorial and mainscreens is excessive and can be difficult to read. Record design is limiting with character fields being fixed length 5 byte fields. A logical record cannot exceed 255 characters although you are allowed up to 49 fields per record.

Documentation: A loose leaf notebook contains 147 pages for all features of Homebase, including the tutorial. The documentation could be more compact if printed on both sides of the sheet, but it is good quality and well organized.

Skill level: While programming skills are certainly not required to use the package, a basic knowledge of computer files is essential in order to properly design file formats. A novice user could, however, learn to use the package fairly quickly due to good documentation which presupposes no expertise.

Reviewer: Norman Garrett

Product Name:	**Parallel Printer Switch**
Equip. Req'd:	Two Centronics-type Printers
Price:	$39.95 plus $2 shipping
Manufacturer:	Ken Branscome Associates 368 N. Walnut Grove Road Midlothian, TX 76065

Description: This piece of hardware consists of a Centronics-type female connector which will plug onto the end of any Centronics parallel interface cable. The "parallel printer switch" allows the user to connect an existing Centronics type printer cable to the printer switch. By using two flat cable extensions, two printers with Centronics type interface can be connected to the printer switch. A switch on the PC card will allow the computer output to be directed to either printer. Thus, if both dot-matrix and letter-quality printers are available, draft copies can be run in dot matrix and final copies with letter quality.

Pluses: This accessory eliminates cable swapping; the design allows easy home-brew flat cables or the option of purchasing standard Radio Shack Model I or Model III cables to interconnect any two printers with standard Centronics interfaces. Two versions are available; one switches the BUSY line and the switches the ACK line, so specify the one you need (check the printer driver protocol of your computer). Ready-made 5' cables are also available at $39.95 each; two cables and the Printer Switch are available for $100 pp.
Minuses: None noted

Documentation: None furnished or needed

Skill level: None

Reviewer: Ralph Tenny

Product Name:	**MMG Form Letter Writer**
Equip. req'd:	Atari 400/800 with 40K disc drive printer
Price:	$29.95
Manufacturer:	MMG Micro Software P.O. Box 131 Marlboro, NJ 07746

Description: *Form Letter Writer* produces letters which can be merged with MMG Data Manager, Mail List, MMG Accounts Receivable, Accounts Payable, MMG Payroll and MMG Inventory programs. This means that owners of these other products now may send form letters to their client base without typing in the names, etc..

Pluses: The program allows printer codes to be entered so it can be configured to work with any printer. The codes can be placed anywhere in the text and will not be printed with the text, but will be sent to the printer. This allows print type to be changed or any special print codes to be sent from within the body of the letter. Form letters can be personalized in the same manner as is done by professional mailers.

Minuses: The text is listed continuously on the screen with an inverse " " symbol marking the end of paragraphs. According to information furnished in the instruction manual, this allows more text to be stored than with the normal format. This special format does take some getting used to. Proportional printer is not supported.

Documentation: The seven page manual explaining use of the program does not cover the interfacing with the programs that it works with as well as it could.

Skill level required: A user with some experience.

Reviewer: Richard E. DeVore

Product Name:	**Decimal Practice-Plato Educational Software**
Equip. req'd:	Atari 400/800/1200XL with 48K disc drive
Price:	?
Manufacturer:	Control Data Publication Co., Inc. P.O. Box 261127 San Diego, CA 92126

Description: *Decimal Practice* is one of the sixteen educational programs in the Plato Educational Software series. As the title implies, the program is designed to teach decimals to elementary math students.

Decimal Practice uses a number line with colored balloons "pinned" at different locations along the line. The object is to estimate where the balloons are positioned on the number line. The student enters an estimate of the location of the balloons from the keyboard and a dart is "thrown" at the balloons. If the aim is good, the balloon bursts and the location is printed. If the dart misses, it sticks in the number line, and the location is printed. This helps learn the relationship between parts and the whole.

The program is divided into two lessons with 8 problems in the first lesson and ten in the second. The problems in the first lesson have whole numbers at each end of the number line and the student can select whether all of the problems should have positive numbers or a mix of positive and negative numbers. The problems in the second lessons are more difficult and have decimal numbers.

Pluses: *Decimals Practice* uses proven teaching methods to instruct students while providing positive feedback by making the student think he is playing a game. The program features a "help" function accessible by typing an "h" instead of a number. When the "h" is pushed the computer will shoot a dart at the number line which will give the student another reference point to use in estimating the location of the balloons.

Minuses: The program was developed using BASIC A+ and loads the language as well as the program so it takes some time to load.

Documentation: There is a forty page booklet furnished with the program. There are sections covering Classroom Strategies, Sample Activities, Student Practice Activities and a Student Record sheet. It is well done and easy to follow.

Skill level required: Elementary School Students.

Reviewer: Richard E. DeVore

Product Name:	**KoalaPad**
Equip. req'd:	Atari Computer with min. 16K, 32K for disk storage disk drive for disk version (tested)
Price:	$99.95
Manufacturer:	Koala Technologies 253 Martens Ave. Mt. View, CA 94040

Description: *The KoalaPad* is a touch tablet designed to be used from joyport 1 of an Atari 400 or 800 computer. It may be operated using your finger or the provided stylus, any other object is not recommended. The unit is small, measuring 6" × 8" × 1" with an active tablet surface area of 4¼" × 4¼". The *KoalaPad* is supplied with a program disk called the "Micro Illustrator". This program along with others soon to be available allow easy use of the touch pad.

Pluses: The touch pad is extremely easy to use with the supplied software. The brief (14 page) owners manual states how to hook the unit to the computer while the 16 page software manual tells how to load the program and use it. In less than 3 minutes a child who had never seen the tablet before had it connected and was drawing on it.

Minuses: Could not find any. It worked exactly as presented.

Documentation: The two manuals supplied with the touch pad and the program disk, while brief, showed clearly how to connect and use the unit.

Skill level required: Beginning computer user.

Reviewer: Richard DeVore

Product Name:	**DataFax**
Equip. req'd:	Apple II or Apple II+
Price:	?
Manufacturer:	Link Systems 1655 26th St. Santa Monica, CA 90404

Description: *DataFax* is a data base management system that runs on the Apple Pascal Operating System. It uses a highly flexible filing process based on keywords within a data record, or folder. The user can select any word or phrase, of variable lengths, to be keys for the folder; they are in turn used to retrieve desired folders for editing or printing.

Pluses: *DataFax* is designed to handle unstructured data, so nearly anything one can type in can be filed and retrieved with ease. Folders are scanned for on single keys or combinations of keys, boolean operators, and wildcard symbols that are easy to work with because they are written in English, not special computer codes. The control keys that function in the Editor may be customized for any system.

Documentation: Over 200 pages of documentation are provided with the *DataFax* package. The manual includes a tutorial section for beginners, a reference section for the basic commands and functions, and Advanced Techniques section for using *DataFax* in conjunction with the USCD Pascal System, and appendices covering hardware requirements, trouble shooting, and a list of all possible error messages and their meanings.

Skill level required: Easy to learn and use for everyone.

Reviewer: John Hedderman

COMPUTER MAIL ORDER

NEC PRINTERS
NEC 2050..................$999.00
NEC 3550..................$1699.00

PERCOM/TANDON DRIVES
5¼" 320K Floppy...........$229.00
5 Meg Hard w/Controller...$1399.00
10 Meg Hard w/Controller...$1699.00
15 Meg Hard w/Controller...$2095.00
20 Meg Hard w/Controller...$2399.00

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

AST RESEARCH
Six Pak Plus...from.........$279.00
Combo Plus II...from.......$279.00
Mega Plus...from...........$309.00
I/O Plus...from.............$139.00

QUADRAM
Quadlink.....................$499.00
Quadboard...as low as.....$289.00
Quad 512 Plus...as low as...$249.00
Quadcolor...as low as......$219.00
Chronograph.................$89.00
Parallel Interface Board.....$89.00
64K RAM Chips Kit...........$59.00

MICROPRO
WordStar/MailMerge.......$349.00
InfoStar.....................$299.00
SpellStar....................$159.00
CalcStar.....................$99.00

MICROSTUF
Crosstalk....................$105.00

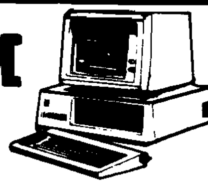
MICROSOFT
Multiplan....................$159.00

ASHTON TATE
dBASE II.....................$389.00
Friday!.......................$185.00

IUS
EasyWriter II................$249.00
EasySpeller..................$119.00
EasyFiler....................$229.00

CONTINENTAL SOFTWARE
1st Class Mail/Form Letter...$79.00
The Home Accnt. Plus.......$88.00

IBM



VISICORP

	IBM	APPLE
VisiCalc		159.00
VisiCalc 4	159.00	
VisiCalc-Advanced		269.00
VisiWord/Spell	249.00	
Visitrend/Plot	199.00	199.00
VisiLink		169.00
VisiFile	199.00	169.00
VisiSchedule	199.00	199.00
Visidex		159.00
VisiPlot		135.00
VisiTerm		75.00
Desktop Plan	199.00	169.00
Bus. Forecast Model	75.00	75.00
Stretch Calc	75.00	75.00
VisiTutor Calc	59.00	59.00
VisiTutor-Advanced	75.00	75.00
VisiTutor Word	59.00	59.00
Vision Calc	249.00	
Vision Graph	129.00	
Vision Mouse	159.00	
Vision Host	319.00	

pfs

	APPLE	IBM
Write:	79.00	89.00
Graph:	79.00	89.00
Report:	79.00	79.00
File:	79.00	89.00
Solutions*: as low as	16.00	16.00

*Call On Titles

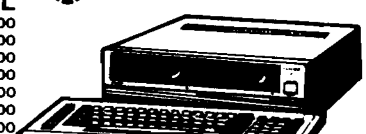
LOTUS
1-2-3.........................$319.00

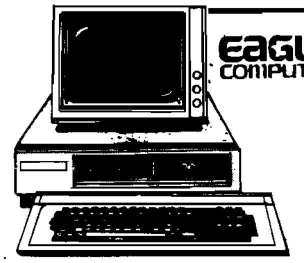
PROFESSIONAL SOFTWARE
PC Plus/The Boss...........$349.00

SYNAPSE
File Manager.................$89.00

MBC-550 PC...CALL
MBC-555 PC...CALL
MBC 1100..................$1499.00
FDD 3200-320K Drive......$389.00
MBC 1150..................$1899.00
MBC 1200..................$1849.00
FDD 6400-640K Drive......$469.00
MBC 1250..................$2099.00
PR 5500 Printer.............$599.00

SANYO





EAGLE COMPUTER
Spirit XL Portable.........$3999.00
IIE-1.........................$1569.00
IIE-2.........................$1899.00
IIE-3.........................$2399.00
IIE-4.........................$3299.00
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

TERMINALS
914...........................$569.00
924...........................$689.00
925...........................$739.00
950...........................$929.00
970...........................$1039.00

COMPUTERS
Teleport Portable.............CALL
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

TeleVideo

PRINTERS

AXIOM
AT-100 Atari Interface......$239.00
CD-100 CBM 64/VIC 20......$239.00
GP-100 Parallel Interface...$199.00

BMC
401 Letter Quality..........$589.00
BX-80 Dot Matrix............$269.00

CENTRONICS
122 Parallel.................$399.00
739-1 Parallel...............$299.00
739-3 Serial.................$349.00

C.ITOH
Gorilla Banana..............$209.00
Prowriter 8510P............$379.00
Prowriter 1550P............$679.00
A10 (18 cps)..................569.00
F10-40.......................$999.00
F10-55......................$1499.00

COMREX
ComWriter II Letter Quality...$499.00

DIABLO
620 Letter Quality..........$949.00
630 Letter Quality.........$1749.00

DAISYWRITER
2000.........................$999.00
Tractor Feed................$109.00

EPSON
MX-80FT, MX-100, RX-80, RX-80FT, FX-80, FX-100...............CALL

IDS
Prism 80...For Configurations...CALL
Prism 32...For Configurations...CALL

MANNESMAN TALLY
160L.........................$589.00
180L.........................$799.00
Spirit 80.....................$309.00

NEC
2010/2030..................$899.00
8023 Dot Matrix............$379.00
8025 Dot Matrix............$669.00
3510 Serial/Letter Quality...$1449.00
3530 Parallel/Letter Quality...$1499.00
7710/7730 Serial/Parallel...$1949.00

OKIDATA
82,83,84,92,93,2350,2410...CALL

SMITH CORONA
TP-2.........................$399.00
Tractor Feed................$119.00

SILVER REED
500 Letter Quality..........$469.00
550 Letter Quality..........$699.00

STAR
Gemini 10X..................$299.00
Gemini P15X.................$399.00
Delta 10.....................$559.00
Serial Board.................$75.00

TOSHIBA............CALL
TRANSTAR..........CALL

PRINTER CABLES
Available for Atari, Commodore, IBM, Apple, Epson, Kaypro, Televideo, Franklin, Eagle, Sanyo, Osborne, NEC, Zenith and many others. We supply all your computer needs!

PAPER SUPPLIES
1000 shts. 8½x11 Tractor Paper...19.99
1000 shts. 14½x11 Tractor Paper..$24.99
1 or 2" Address Labels........$9.99

MONITORS

AMDEK
300 Green...................$149.00
300 Amber...................$159.00
310 Amber...................$169.00
Color 1......................$279.00
Color 1 Plus.................$299.00
Color 2......................$399.00
Color 2 Plus.................$419.00
Color 3......................$349.00
Color 4......................$699.00

BMC
12" Green....................$79.99
12" Green HI-RES...........$119.99
9191-13" Color.............$249.99

GORILLA
12" Green....................$88.99
12" Amber....................$95.99

NEC
JB 1260 Green..............$109.99
JB 1201 Green..............$149.99
JB 1205 Amber..............$159.99
JC 1215 Color...............$299.99
JC 1216 RGB.................$429.99

PRINCETON GRAPHICS
HX-12 RGB...................$519.00

SAKATA
100..........................$269.00

TAXAN
12" Green...................$119.00
12" Amber...................$129.00
Taxan 1 RGB.................$279.00
Taxan 210...................$299.00

USI
Pi 1, 9" Green...............$99.99
Pi 2, 12" Green.............$119.99
Pi 3, 12" Amber.............$149.99
Pi 4, 9" Amber..............$139.99
1400 Color..................$269.99

ZENITH
ZVM 122 Amber..............$109.00
ZVM 123 Green...............$89.99
ZVM 135 Color/RGB.........$469.99

MODEMS

ANCHOR
Mark I (RS-232)..............$79.00
Mark II (Atari)...............$79.00
Mark III (TI-99).............$109.00
Mark IV (CBM/PET)..........$125.00
Mark V (Osborne).............$95.00
Mark VI (IBM-PC)............$169.00
Mark VII (Auto Ans/Auto Dial)...$119.00
Mark XII (1200 Baud)........$299.00
TRS-80 Color Computer.....$99.00
9 Volt Power Supply...........$9.00

HAYES
Smartmodem 300.............$219.00
Smartmodem 1200............$509.00
Smartmodem 1200B..........$459.00
Micromodem II...............$265.00
Micromodem II Plus.........$299.00
Micromodem IIE..............$269.00
Micromodem 100.............$299.00
Smart Com II.................$89.00
Chronograph.................$199.00

NOVATION
J-Cat........................$99.99
SmartCat 103................$179.00
SmartCat 103/212...........$399.00
AutoCat.....................$219.00
212 AutoCat.................$549.00
Apple Cat II.................$249.00
212 Apple Cat...............$569.00
Apple Cat 212 Upgrade......$309.00
Cat..........................$139.99
D-Cat.......................$149.00
PC-Cat......................$339.00

ZENITH
ZT-1........................$309.00
ZT-10.......................$339.00
ZT-11.......................$369.00

APPLE INTERFACE CARDS & BUFFERS
Choose from PKASO, Orange Micro, MPC, MicroMax, Tymac, Quadram & Practical Peripherals...........CALL

WEST
800-648-3311
In NV call (702)588-5654. Dept. 0315
Order Status Number: 588-5654
P.O. Box 6689, Stateline, NV 89449

CANADA
800-268-4559
In Toronto call (416)828-0866. Dept 0315
Order Status Number: 828-0866
2505 Dunwin Drive, Unit 1B
Mississauga, Ontario, Canada L5L1T1

EAST
800-233-8950
In PA call (717)327-9575. Dept 0315
Order Status Number: 327-9576
Customer Service Number: 327-1450
477 E. Third St., Williamsport, PA 17701

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 new catalog.

COMPUTER MAIL ORDER

APPLE/FRANKLIN DISK DRIVES

MICRO-SCI

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

RANA

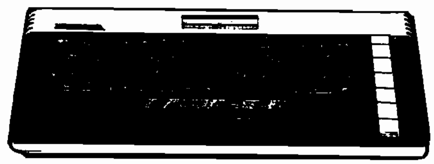
Elite 1 ..... $279.00
Elite 2 ..... $389.00
Elite 3 ..... $569.00

APPLE IIe STARTER PACK
64K Apple IIe, Disk Drive & Controller, 80 Column Card, Monitor II & DOS 3.3
COMPLETE $1199.00

FRANKLIN

ACE 1000 Color Computer ..... CALL
ACE Family Pack System ..... CALL
ACE PRO PLUS System ..... CALL
ACE 1200 Office Mgmt. System ..... CALL
"NOT IIE EXPENSIVE"

ATARI HOME COMPUTERS



600XL $189
800XL $299
1200XL ..... CALL
1400XL ..... CALL

1010 Recorder $74.00
1020 Color Printer $249.00
1025 Dot Matrix Printer $449.00
1027 Letter Quality $299.00
1030 Direct Connect Modem ..... CALL
1050 Disk Drive $379.00
CX30 Paddle $12.00
CX40 Joystick ..... each ... $8.00
CX77 Touch Tablet $64.00
CX80 Trak Ball $48.00
CX85 Keypad $105.00
488 Communicator II $229.00
4003 Assorted Education $47.00
4011 Star Raiders $33.00
4012 Missile Command $29.00
4013 Asteroids $29.00
5049 VisiCalc $159.00
7097 Logo $79.00
7101 Entertainer $69.00
7102 Arcade Champ $75.00
8026 Dig Dug $33.00
8030 E.T. Phone Home $33.00
8031 Donkey Kong $39.00
8033 Robotron $35.00
8034 Pole Position $39.00
8036 Atari Writer $79.00
8040 Donkey Kong, Jr. $39.00
8043 Ms. PacMan $39.00
8044 Joust $39.00

PERCOM

AT 88-S1 ..... $369.00
AT 88-A2 ..... $259.00
AT 88-S2 ..... $529.00
AT 88-S1PD $429.00
AT 88-DDA $119.00
RFD 40-S1 ..... $449.00
RFD 40-A1 ..... $269.00
RFD 40-S2 ..... $699.00
RFD 44-S1 ..... $539.00
RFD 44-S2 ..... $869.00

TEXAS INSTRUMENTS

TX 99-S1 ..... $279.00

RANA

1000 ..... $319.00

TRAK

AT-D2 ..... $389.00

MEMORY BOARDS

Axlon 32K $59.00
Axlon 48K $99.00
Axlon 128K $299.00
Intec 32K $59.00
Intec 48K $85.00
Intec 64K $99.00
Intec Real Time Clock $29.00

ALIEN VOICE BOX

Atari $119.00
Apple $149.00

KOALA PAD

Atari $75.00
Apple $85.00
IBM $95.00
CBM 64 ..... $75.00

commodore

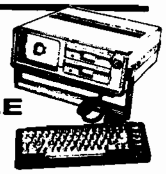
CBM 8023 ... $599

CBM 4032 ..... $599.00
CBM 8096 ..... $869.00
CBM 9000 ..... $999.00
B128-80 ..... $769.00
CBM 64K Memory Board $269.00
8032 to 9000 Upgrade $269.00
2031LP Disk Drive $299.00
8050 Disk Drive $949.00
8250 Disk Drive $1199.00
4023 Printer $379.00
8023 Printer $569.00
6400 Printer $1399.00
Z-RAM $499.00
Silicon Office $699.00
The Manager $199.00
Soft ROM $125.00
VisiCalc $159.00

PROFESSIONAL SOFTWARE

Word Pro 2 Plus $159.00
Word Pro 3 Plus $189.00
Word Pro 4 Plus/5 Plus ... each ... $279.00
InfoPro $179.00
Administrator $399.00
Power $79.00

SX-64 PORTABLE $869.00



VIC 20 ..... CALL
CBM 64 ..... $199

MSD SD1 Disk Drive (C-64) $349.00
C1541 Disk Drive $249.00
C1530 Datasette $69.00
C1520 Color Printer/Plotter $169.00
M-801 Dot Matrix Printer $219.00
C1526 Dot Matrix/Serial $279.00
C1702 Color Monitor $249.00
C1311 Joystick $4.99
C1312 Paddles $11.99
C1600 VIC Modem $59.00
C1650 Auto Modem $89.00
Logo 64 ..... $49.00
Pilot 64 ..... $39.00
Simon's Basic $19.00
Word Pro 64 Plus $59.00
Parallel Printer Interface $49.00
Calc Result 64 ..... $129.00
Codewriter 64 ..... $75.00
Quick Brown Fox $49.00
Word Pro 64 Plus $59.00

We stock a full inventory of software for Commodore, such as: Artworx, Broderbund, Commercial Data, Creative Software, Epyx, HES, MicroSpec, Nufekop, Romox, Sirius, Synapse, Thorn EMI, Tronix, UMI, Victory, Spinnaker, Rainbow & Timeworks!
CALL FOR DETAILS!

DISKETTES

MAXELL

5¼" MD-1 ..... $29.00
5¼" MD-2 ..... $39.00
8" FD-1 (SS/DD) $39.00
8" FD-2 (DS/DD) $49.00

VERBATIM

5¼" SS/DD $26.99
5¼" DS/DD $36.99

ELEPHANT

5¼" SS/SD $18.49
5¼" SS/DD $22.99
5¼" DS/DD $28.99

HEAD

5¼" Disk Head Cleaner $14.99

DISK HOLDERS

INNOVATIVE CONCEPTS

Flip-n-File 10 ..... $3.99
Flip-n-File 50 ..... $17.99
Flip-n-File (400/800 ROM) Holder $17.99

LJK ENTERPRISES

Atari Letter Perfect-Disk (40/80) $79.99
Atari Letter Perfect-ROM (40 col) $79.99
Atari Letter Perfect-ROM (80 col) $79.99
Atari Data Perfect-ROM (80 col) $79.99
Atari Spell Perfect-DISK $59.99
Atari Utility/MailMerge $21.00
Apple Letter Perfect $99.00
Apple Data Perfect $75.00
Apple LJK Utility $21.00
Apple Lower Case Generator $19.00

CONTROLLERS & JOYSTICKS

WICO

Joystick $21.99
3-way Joystick $22.99
Famous Red Ball $23.99
Power Grip $21.99
BOSS Joystick $17.99
ATARI/VIC Trak Ball $34.99
Apple Trak Ball $54.99
Apple Adapter $15.99
Apple Analog $37.99

KRAFT

Joystick $41.99
Atari Single Fire $12.99
Atari Switch Hitter $15.99
Apple Paddles $34.99
IBM Paddles $34.99
IBM Joystick $46.99

AMIGA

3100 Single $13.99
3101 Pair $19.99
Joyboard $37.99

TG

Atari Trak Ball $47.99
Apple Joystick $47.99
Apple Trak Ball $47.99

HANDHELD COMPUTERS

hp

41CX $249.99
41CV $199.99
41C $144.99
HP 10C $51.99
HP 11C $69.99
HP 12C $88.99
HP 15C $88.99
HP 16C $88.99
HP 75C $749.99
HPIL Module $98.99
HPIL Cass. or Printer $359.99
Card Reader $143.99
Extended Function Module $63.99
Time Module $63.99

NEC

PC-8201 Personal Computer $599.00
PC-8221A Thermal Printer $149.00
PC-8281A Data Recorder $99.00
PC-8201-06 8K RAM Chips $105.00
PC-8206A 32K RAM Cartridge $329.00

SHARP

PC-1500A ... $165.99
PC-1250A ... $88.99

CE-125 Printer/Cassette $128.99
CE-150 Color Printer/Cassette $171.99
CE-155 8K RAM $93.99
CE 161 16K RAM $134.99
CE 500 ROM Library $29.99

TIMEX/SINCLAIR

Timex/Sinclair 1000 ..... $24.99
Timex/Sinclair 2086 ..... CALL
16K Memory $25.00
2040 Printer $99.99
VuCalc $17.99
Mindware Printer $99.99

WEST
800-648-3311
In NV call (702)588-5654. Dept 0315
Order Status Number: 588-5654
P.O. Box 6689, Stateline, NV 89449

CANADA
800-268-4559
In Toronto call (416)828-0866. Dept 0315
Order Status Number: 828-0866
2505 Dunwin Drive, Unit 1B
Mississauga, Ontario, Canada L5L1T1

EAST
800-233-8950
In PA call (717)327-9575. Dept 0315
Order Status Number: 327-9576
Customer Service Number: 327-1450
477 E. Third St., Williamsport, PA 17701

CANADIAN ORDERS: All prices are subject to shipping, tax and currency fluctuations. Call for exact pricing in Canada.
INTERNATIONAL ORDERS: All shipments outside the continental United States must be pre-paid by certified check only. Include 3% (minimum $5.00) shipping and handling. EDUCATIONAL DISCOUNTS: Additional discounts are available to qualified Educational Institutions.
APO & FPO: Add 3% (minimum $5.00) shipping and handling.

Least-Squares Curve Fitter

by Brian Flynn

Plot and depict the apparent trend between variables (such as stocks and interest rates) with the statistical routine

The urge is almost irresistible. You see a plot of points between two variables, such as incidents of heart disease and frequency of cigarette smoking, or wheat harvest and yearly rainfall, or stock prices and interest rates. And you want to draw a line through the points to depict the apparent trend, as Figure 1 shows. Least-Squares Curve Fitter is a statistical routine which will enable you to satisfy your desire for a line in a wide range of circumstances. More technically, Curve Fitter estimates a multiple linear regression equation in Apple II Basic. With little modification, the program will run on non-Apple systems as well. This article will explain the use of Curve Fitter by presenting a real-world example. Regression statistics will then be interpreted.

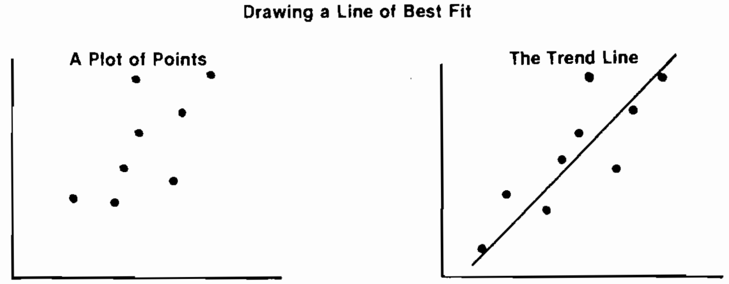


Figure 1

A Real-World Example

According to many Wall Street gurus, the only sure thing about the stock market is that it will fluctuate. The only certainty, in other words, is change. Nevertheless, is it not possible to devise an investment strategy that will work successfully on average, and over the long haul? With painstaking work and steady nerves, is it not possible to tilt the merciless roulette wheel of Wall Street in our favor for once?

Perhaps it is. Many of us have probably noted that stock prices tend to fall when interest rates rise, and conversely, that stock prices tend to rise when interest rates fall. In short, the two variables seem inversely related. When one goes up the other goes down, and vice versa.

To test our hypothesis about stock prices and interest rates, we first gather the observations shown in Table 1, and then run Curve Fitter. The computer soon displays

The Maximum Allowable Numbers of Observations and Explanatory Variables Are:

Observations = 50
Exp. Variables = 6

Change the Values in Line 2020 for Different Limits

"Stock Prices" is called the dependent variable, or Y. Our goal is to explain changes in Standard and Poor's Index of 500 Leading Stocks from January 1982 to June 1983, or for 18 months in all. The term to do the explaining is called, logically enough, the explanatory variable, or X. In our case, we have only one X, namely "Interest Rates."

The computer now asks us to enter our data. First comes the dependent variable. Starting with January 1982, we key-in 100.0 for Y(1), 97.6 for Y(2), and so on down the list, all the way to 141.9 for Y(18), or June 1983. When the computer asks for Y(19), we simply hit RETURN without entering a number beforehand. This tells the computer that we have 18 observations. Data on interest rates are entered similarly.

After we have entered the values of Table 1, the computer displays what we have keyed-in, and gives us a

Table 1

Stock Prices and Interest Rates

Year and Month	S & P 500 Stock Index	3 Month T-Bill Rate
82:1	100.0	12.3%
:2	97.6	13.5
:3	94.5	12.7
:4	99.2	12.7
:5	99.2	12.1
:6	93.5	12.5
:7	93.3	11.4
:8	93.5	8.7
:9	104.4	7.9
:10	113.1	7.7
:11	117.8	8.1
:12	118.8	7.9
83:1	123.0	7.9
:2	125.2	8.1
:3	129.5	8.4
:4	134.5	8.2
:5	139.9	8.2
:6	141.9	8.8

chance to make corrections. Ten observations are shown at a time on the screen, so do not worry about scrolling.

The computer now estimates our regression equation, and then displays

Regression Results

Term	Value	t-Statistic
B0	165.945	11.996
B1	-5.467	-3.979

R-Squared = 0.497
F-Statistic = 15.830

Standard Error of the Estimate = 12.455

Durbin-Watson Statistic = 0.263

These statistics are interpreted as follows. First, B0 is the Y-intercept of our equation and B1 the slope, as Figure 2 illustrates. The Y-intercept of 165.9 means that, if interest rates were zero, our index of stock prices would equal 165.9, or so we estimate. The slope of -5.5 means that a one percentage point rise in interest rates will induce an estimated 5.5 unit drop in the Index of 500 Leading Stocks. In short, the relationship between stock prices and interest rates is indeed negative, as conjectured.

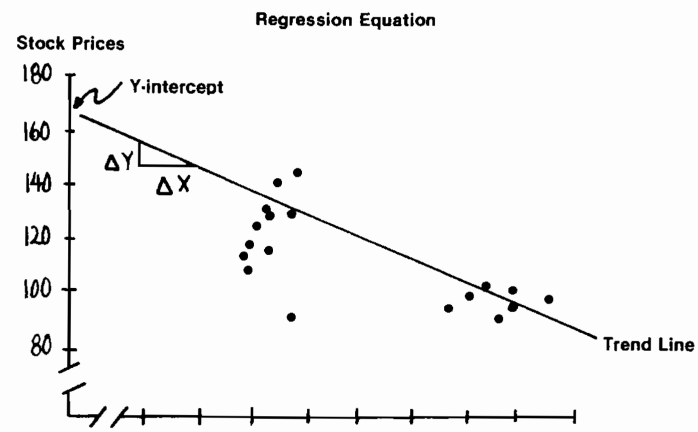


Figure 2. The slope of a line is the change in Y (denoted by ΔY) divided by the change in X (denoted by ΔX).

These values of B0 and B1 are merely best guesses rather than perfect measurements, however. The true values are always unknown and must be estimated. But this, after all, is the purpose of regression analysis.

The t-statistics indicate how precise the estimates of B0 and B1 are. As a rough rule of thumb, a t-value greater than two in absolute value means that an explanatory variable is statistically significant in explaining changes in Y.

The next three values are goodness-of-fit statistics. The R-squared, also called the coefficient of determination, is the proportion of variation in the dependent variable explained by the regression equation. It ranges from 0 to 1, with a value close to 0 meaning that the equation fits the data poorly, and with a value close to 1 meaning that it fits the data well. Figure 3 illustrates this. The R-squared of 0.497 in our example means that changes in interest rates explain roughly 50% of the total variation in stock prices. The source of the other 50% of the variation is unfortunately unknown.

Next, the F-statistic is the ratio of the explained to the unexplained variance in Y. The higher the value of F, the better does the regression equation explain changes in the dependent variable. The standard error of the estimate is a measure of the average error made in predicting Y using the regression equation, or 12.5 index points in our example.

Finally, the Durbin-Watson statistic is used in testing for first-order serial correlation among regression residuals. A residual, let me hasten to explain, is an actual value of Y minus the corresponding value of Y predicted by the regression equation, as Figure 4 shows. As a rough rule of thumb, a DW value of around 2 means that serial correlation is not a problem. The miserly value of 0.263 in our example warns us that some systematic variation in stock prices is unexplained by interest rates.

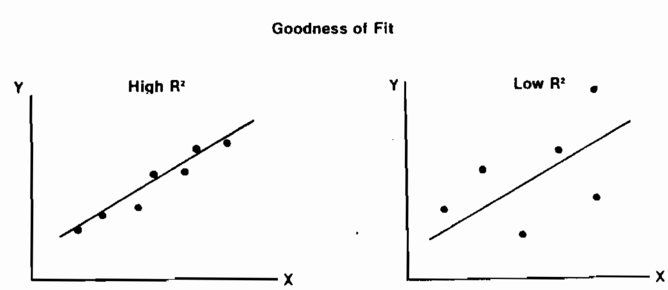


Figure 3. A high R^2 means that the regression line fits the data well. A low R^2 means that the regression line fits the data poorly.

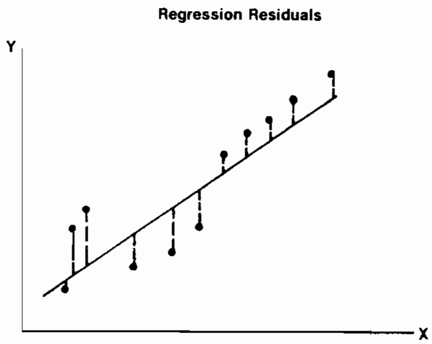


Figure 4. A residual is the vertical distance between an actual value of Y and the estimated regression line.

Summary

In summary, our regression results are only fair. Changes in interest rates account for roughly half of the fluctuation in stock prices over the last 18 months. A large part of the market's movement, then, is left unexplained. Hence, trying to predict the future course of the stock market using interest rates alone is a risky business indeed. Perhaps Madame Zelna's crystal ball can defeat the dark forces of ignorance and uncertainty, and shed light on the problem.

```
1 REM  FLYNN MARCH 1984
  10  REM  MULTIPLE LINEAR REGRESSION
  20  REM  BRIAN J. FLYNN
  30  REM  NOVEMBER 1983
  40  REM  INTIALIZE
  50  GOSUB 1000
  60  REM  ENTER & EDIT DATA
  70  GOSUB 3000
  80  REM  COMPUTE
  90  GOSUB 7000
 100  REM  DISPLAY RESULTS
 110  GOSUB 12500
 120  END
********************************
*  INSERT COMPUTER SPECIFIC    *
*        DRIVERS HERE          *
*  (SEE TABLE OF SUBROUTINES)  *
********************************
 1000  REM  INITIALIZE
 1010  REM  HEADING
 1020  GOSUB 1500
 1030  REM  INITIAL VALUES
 1040  GOSUB 2000
 1050  REM  INTRODUCTION
 1060  GOSUB 2500
 1070  RETURN
 1500  REM  HEADING
 1510  GOSUB 300
 1520  VT=11:HT=15: GOSUB 400: PRINT "MULTIPLE"
 1530  VT=12:HT=16: GOSUB 400: PRINT "LINEAR"
 1540  VT=13:HT=17: GOSUB 400: PRINT "REGRESSION"
 1550  FOR D=1 TO 750: NEXT D
 1560  RETURN
 2000  REM  INITIAL VALUES
 2010  REM  MAX NUMBER OF OBSERVATIONS & X'S
 2020  DATA  50,6
 2030  READ NX,KX
 2040  PX=KX+1
 2050  DIM C(PX),X(NX,PX),R(PX,2*PX),E(NX),B$(KX),V$(KX)
 2055  DIM T(PX),B(PX)
 2060  REM  COEFFICIENT SYMBOLS
 2070  FOR I=0 TO KX
 2080  B$(I)="B"+STR$(I)
 2090  NEXT I
 2100  REM  VARIABLE SYMBOLS
 2110  V$(0)="Y"
 2120  FOR I=1 TO KX
 2130  V$(I)="X"+STR$(I)
 2140  NEXT I
 2150  RETURN
 2500  REM  INTRODUCTION
 2510  GOSUB 300
 2520  PRINT "THIS PROGRAM ESTIMATES A MULTIPLE"
 2530  PRINT "LINEAR REGRESSION EQUATION."
 2540  PRINT
 2550  PRINT "THE MAXIMUM ALLOWABLE NUMBERS OF"
 2560  PRINT "OBSERVATIONS & EXPLANATORY VARIABLES"
 2570  PRINT "ARE:"
 2580  PRINT
 2590  PRINT "         OBSERVATIONS=";NX
 2595  PRINT "EXPLANATORY VARIABLES=";KX
 2600  PRINT
 2610  PRINT "CHANGE THE VALUES IN LINE 2020"
 2620  PRINT "FOR DIFFERENT LIMITS."
 2630  VT=22:HT=6: GOSUB 400
 2640  PRINT "HIT ANY KEY TO CONTINUE ";
 2650  GOSUB 600: Z$=XX$
 2660  RETURN
 3000  REM  ENTER & EDIT DATA
 3010  REM  NUMBER OF X'S
 3015  GOSUB 3250
 3020  REM  DATA ON Y
 3030  GOSUB 3500
```

```
3040  REM  DATA ON THE X'S
3050  GOSUB 4000
3060  REM  EDIT
3070  GOSUB 4500
3080  RETURN
3250  REM  NUMBER OF X'S
3260  GOSUB 300
3270  VT=1:HT=1: GOSUB 400
3280  PRINT "HOW MANY EXPLANATORY VARIABLES"
3290  PRINT "ARE IN YOUR REGRESSION EQUATION,"
3300  PRINT "CONSTANT TERM EXCLUDED ?";
3310  GOSUB 600: K$=XX$
3320  K=VAL(K$)
3330  REM  CHECK FOR LEGAL NUMBER
3340  IF K>0 AND K<=KX THEN 3400
3350  VT=22:HT=7: GOSUB 400
3360  IF K<1 THEN PRINT "AT LEAST ONE X IS NEEDED ! "
3370  IF K>KX THEN PRINT "SORRY, ONLY ";KX;
      " X'S ALLOWED !"
3380  FOR D=1 TO 5: GOSUB 800: NEXT D
3390  GOTO 3270
3400  RETURN
3500  REM  DATA ON Y
3510  GOSUB 300
3520  PRINT "PLEASE ENTER DATA ON THE DEPENDENT"
3530  PRINT "VARIABLE, OR Y.  HIT 'RETURN'"
3540  PRINT "WHEN THROUGH."
3550  N=NX
3560  FOR I=1 TO NX
3570  VT=5:HT=10:SP=15: GOSUB 500
3580  VT=5:HT=2: GOSUB 400: PRINT "Y( ";I; TAB(8);")= ";
3590  GOSUB 700: Z$=XX$
3600  IF Z$="" THEN N=I-1:I=NX: GOTO 3620
3610  X(I,0)=VAL(Z$)
3620  NEXT I
3630  IF N>2 THEN 3660
3640  VT=22:HT=5: GOSUB 400
3645  PRINT "AT LEAST 3 OBSERVATIONS NEEDED !"
3650  FOR D=1 TO 20: GOSUB 800: NEXT D
3655  GOTO 3510
3660  RETURN
4000  REM  DATA ON THE X'S
4010  FOR I=1 TO K
4020  GOSUB 300
4030  PRINT "PLEASE ENTER DATA FOR ";V$(I);":"
4040  FOR J=1 TO N
4050  VT=5:HT=10:SP=20: GOSUB 500
4060  VT=5:HT=1: GOSUB 400
4065  PRINT V$(I);"( ";J; TAB(8);")= ";
4070  GOSUB 700: Z$=XX$
4080  X(J,I)=VAL(Z$)
4090  NEXT J: NEXT I
4100  RETURN
4500  REM  EDIT DATA
4510  FOR I=0 TO K
4520  FOR L=0 TO INT((N-1)/10)
4530  REM  DISPLAY DATA
4540  GOSUB 5000
4550  REM  CORRECT DATA
4560  GOSUB 5500
4570  NEXT L: NEXT I
4580  RETURN
5000  REM  DISPLAY DATA
5010  REM  DISPLAY UP TO 10 OBSERVATIONS AT A TIME
5020  GOSUB 300
5030  PRINT "THESE ARE VALUES OF ";V$(I);":"
5040  PRINT
5050  FOR J=1 TO 10
5060  IF J+L*10<=N THEN PRINT V$(I);"( ";J+L*10;TAB(8);
      ")=";X(J+L*10,I)
5070  NEXT J
5080  RETURN
5500  REM  CORRECT DATA
5510  GOSUB 800
5520  VT=19:HT=1: GOSUB 400: PRINT "CORRECTIONS(Y/N) ";
5530  GOSUB 600: A$=XX$
5540  IF A$="N" THEN 5670
5550  IF A$<>"Y" THEN 5510
5560  VT=21:HT=18:SP=20: GOSUB 500
5565  VT=22:HT=18:SP=20: GOSUB 500
5567  PRINT CHR$(BL)
5570  VT=22:HT=1: GOSUB 400: PRINT "TO BE CORRECTED ";
5590  GOSUB 700: S$=XX$
5600  Q=INT(VAL(S$))
5610  IF Q<(1+L*10) OR Q>N OR Q>(10+L*10) THEN GOTO 5700
5620  VT=24:HT=1: GOSUB 400:
      PRINT "WHAT SHOULD THE VALUE BE ";
5630  GOSUB 800
5640  GOSUB 700: S$=XX$
5650  X(Q,I)=VAL(S$)
5660  GOSUB 5000: GOTO 5510
5670  RETURN
5700  VT=22:HT=18: GOSUB 400: PRINT " OUT OF BOUNDS !"
5710  FOR D=1 TO 1000: NEXT D: GOTO 5565
7000  REM  COMPUTE
7010  REM  DEGREES OF FREEDOM
7020  GOSUB 7500
7025  REM  INSERT VECTOR OF 1'S FOR CONSTANT TERM
7027  GOSUB 7750
7030  REM  TALLY MATRIX OF CROSS PRODUCTS
7040  GOSUB 8000
7050  REM  INVERT MATRIX
7060  GOSUB 8500
7070  REM  COMPUTE COEFFICIENTS
7080  GOSUB 9000
7090  REM  COMPUTE ANOVA STATISTICS
7100  GOSUB 9500
7110  REM  COMPUTE T-STATISTICS
7120  GOSUB 12000
7130  RETURN
7500  REM  DEGREES OF FREEDOM
7510  V=N-K-1
7520  GOSUB 300
7530  IF V<1 THEN GOTO 7600
7540  RETURN
7600  PRINT "YOU HAVE ONLY ";V;" DEGREES OF FREEDOM !":
      STOP
7610  RETURN
7750  REM  VECTOR OF 1'S
7760  REM  MAKE ROOM
7770  FOR I=K TO 1 STEP -1
7780  FOR J=1 TO N
7790  X(J,I+1)=X(J,I)
7800  NEXT J: NEXT I
7810  REM  INSERT
7820  FOR J=1 TO N
7830  X(J,1)=1
7840  NEXT J
```

```
7850  P=K+1
7860  RETURN
8000  REM  MATRIX OF CROSS PRODUCTS
8010  VT=12:HT=0: GOSUB 400: PRINT " COMPUTING ..."
8020  FOR I=1 TO P
8030  FOR J=1 TO P
8040  R(I,J)=0
8050  FOR L=1 TO N
8060  R(I,J)=R(I,J)+X(L,I)*X(L,J)
8070  NEXT L: NEXT J: NEXT I
8080  RETURN
8500  REM  INVERT MATRIX
8510  REM  TACK ON IDENTITY MATRIX
8520  FOR I=1 TO P
8530  FOR J=1 TO P
8540  IF I=J THEN R(I,J+P)=1
8545  IF I<>J THEN R(I,J+P)=0
8550  NEXT J: NEXT I
8560  REM  INVERT
8570  FOR I=1 TO P
8580  REM  ADJUST KEY ROW
8590  C=R(I,I)
8600  FOR J=I TO 2*P
8610  R(I,J)=R(I,J)/C
8620  NEXT J
8630  REM  ADJUST REMAINING ROWS
8640  FOR J=1 TO P
8650  X=R(J,I)
8660  FOR L=I TO 2*P
8670  IF J<>I THEN R(J,L)=R(J,L)-X*R(I,L)
8680  NEXT L:NEXT J: NEXT I
8690  RETURN
9000  REM  TALLY COEFFICIENTS
9005  REM  X'Y VECTOR
9010  FOR I=1 TO P
9020  C(I)=0
9030  FOR J=1 TO N
9040  C(I)=C(I)+X(J,I)*X(J,0)
9050  NEXT J: NEXT I
9060  REM  COEFFICIENTS
9070  FOR I=1 TO P
9080  B(I)=0
9090  FOR J=1 TO P
9100  B(I)=B(I)+R(I,J+P)*C(J)
9110  NEXT J: NEXT I
9120  RETURN
9500  REM  ANOVA STATISTICS
9510  REM  RESIDUAL VARIANCE
9520  GOSUB 10000
9530  REM  SUMMARY STATISTICS
9540  GOSUB 10500
9550  REM  DURBIN-WATSON STATISTIC
9560  GOSUB 11500
9570  RETURN
10000 REM  RESIDUAL VARIANCE
10010 REM  VECTOR OF RESIDUALS
10020 FOR I=1 TO N
10030 YH=0
10040 FOR J=1 TO P
10050 YH=YH+X(I,J)*B(J)
10060 NEXT J
10065 E(I)=X(I,0)-YH
10067 NEXT I
10070 REM  ERROR SUM OF SQUARES
10080 ES=0
10090 FOR I=1 TO N
10100 ES=ES+E(I)*E(I)
10110 NEXT I
10120 REM  RESIDUAL VARIANCE
10130 RV=ES/V
10140 RETURN
10500 REM  ANOVA TERMS
10510 REM  TOTAL SUM OF SQUARES
10520 GOSUB 11000
10530 REM  REGRESSION SUM OF SQUARES
10540 RS=TS-ES
10550 REM  STANDARD ERROR OF THE ESTIMATE
10560 SE=SQR(RV)
10570 REM  F-STATISTIC
10580 F=(RS/K)/RV
10590 REM  R-SQUARED
10600 RQ=RS/TS
10610 RETURN
11000 REM  TOTAL SUM OF SQUARES
11010 S=0:SQ=0
11020 FOR I=1 TO N
11030 S=S+X(I,0)
11040 SQ=SQ+X(I,0)^2
11050 NEXT I
11060 TS=SQ-S*S/N
11070 RETURN
11500 REM  DURBIN-WATSON STATISTIC
11510 REM  NUMERATOR
11520 S=0
11530 FOR I=2 TO N
11540 S=S+(E(I)-E(I-1))^2
11550 NEXT I
11560 REM  VALUE
11570 DW=S/ES
11580 RETURN
12000 REM  T-STATISTICS
12010 FOR I=1 TO P
12020 T(I)=B(I)/SQR(RV*R(I,I+P))
12030 NEXT I
12040 RETURN
12500 REM  DISPLAY RESULTS
12510 REM  EQUATION
12520 GOSUB 13000
12530 REM  SUMMARY STATISTICS
12540 GOSUB 13500
12550 RETURN
13000 REM  EQUATION
13010 GOSUB 300
13020 PRINT TAB(9)"REGRESSION RESULTS"
13030 PRINT
13040 PRINT "TERM"; TAB(12)"VALUE";
     TAB(24)"T-STATISTIC"
13050 PRINT "----"; TAB(12)"-----";
     TAB(24)"-----------"
13060 PRINT
13070 FOR I=1 TO P
13080 PRINT B$(I-1); TAB(9)B(I); TAB(24)T(I)
13090 NEXT I
13100 VT=22:HT=6: GOSUB 400
13110 PRINT "HIT ANY KEY TO CONTINUE ";
13120 GOSUB 600: Z$=XX$
```

```
13130  RETURN
13500  REM  SUMMARY STATISTICS
13510  GOSUB 300
13520  PRINT TAB(8)"SUMMARY STATISTICS"
13530  PRINT
13540  PRINT "R-SQUARED       =";RQ
13545  PRINT
13550  PRINT "F-STATISTIC     =";F
13560  PRINT
13570  PRINT "STANDARD ERROR"
13580  PRINT "OF THE ESTIMATE=";SE
13590  PRINT
13600  PRINT "DURBIN-WATSON"
13610  PRINT "STATISTIC       =";DW
13620  VT=22:HT=6: GOSUB 400
13630  PRINT "HIT ANY KEY TO CONTINUE ";
13640  GOSUB 600: Z$=XX$
13650  RETURN
```

Listing Notes

The above listing does **not** include routines to position the cursor, get character input, input strings or make a sound. These are provided below for three BASIC's: Flex, Applesoft and Commodore 64. Key in the appropriate version for your microcomputer. If you have some other micro, except for the Atari, you should be able to adapt this program by fixing up these I/O routines to match the capabilities/limitations of your system.

If you have an Atari, the program will require more extensive changes than just these I/O routines. This is because the program makes use of string arrays which are not simply supported on the Atari.

```
200  REM   APPLE II SUBROUTINES

299  REM  ** HOME AND CLEAR DISPLAY **
300  HOME : RETURN

399  REM  ** POSITION CURSOR **
400  IF VT>0 THEN VTAB(VT)
410  IF HT>0 THEN HTAB(HT)
420  RETURN

499  REM  ** POSITION CURSOR AND PRINT SPACES **
500  GOSUB 400: PRINT SPC(SP);: RETURN

599  REM  ** GET SUBROUTINE **
600  GET XX$: RETURN

699  REM  ** INPUT SUBROUTINE **
700  INPUT XX$: RETURN

799  REM  ** MAKE SOUND **
800  PRINT CHR$(7);: RETURN
```

```
200 REM   COMMODORE SUBROUTINES

299 REM   ** HOME AND CLEAR DISPLAY **
300 PRINT "{CLEAR}";:RETURN

399 REM   ** POSITION CURSOR **
400 PRINT "{HOME}";
410 FOR XX=1 TO VT:PRINT :NEXT XX
420 IF HT>0 THEN PRINT TAB(HT);
430 RETURN

499 REM  ** POSITION CURSOR AND SPACE **
500 GOSUB 400: PRINT SPC(SP);: RETURN
599 REM  ** GET SUBROUTINE **
600 XX$=""
610 GET XX$: IF XX$="" THEN 610
620 RETURN

699 REM  ** INPUT SUBROUTINE **
700 XX$="":PRINT "{SPACE10,LEFT10}";:INPUT XX$:
    RETURN

799 REM  ** MAKE SOUND (OPTIONAL) **
800 RETURN : REM  ADD CODE TO MAKE A
801 REM  SOUND IF YOU SO DESIRE !!!
```

```
200  REM   FLEX SUBROUTINES

299  REM   ** CLEAR DISPLAY **
300  PRINT CHR$(11);CHR$(27);"X";CHR$(24);:RETURN

399  REM   ** POSITION CURSOR **
400  IF VT>0 THEN PRINT CHR$(11);:FOR II=1 TO VT:PRINT:
    NEXT II
410  IF HT>0 THEN PRINT TAB(HT);
420  RETURN

499  REM   ** POSITION CURSOR AND SPACE **
500  GOSUB 400: PRINT SPC(SP);: RETURN

599  REM   ** GET CHARACTER ROUTINE **
600  INPUT XX$: IF XX$="X" THEN XX$=""
610  RETURN

699  REM   ** INPUT ROUTINE **
700  GOTO 600

799  REM  ** MAKE SOUND (OPTIONAL) **
800  RETURN : REM ADD CODE HERE TO MAKE A
801  REM  SOUND IF YOU SO DESIRE !!
```

Brian Flynn may be reached at Flynn Laboratories, 1704 Drewlaine Drive, Vienna, VA 22180.

[Ed. Note: It is interesting to examine the differences between the various implementations of BASIC on as fundamental an operation as **INPUT***. To* INPUT *a* **NULL** *string in Applesoft BASIC, a simple* **INPUT XX$** *will suffice. In Commodore BASIC you must first set the string to the null string by a* **XX$** = "". *The screen location under the cursor must be a space or the character under the cursor will be returned as the string. In Flex BASIC, used on our FOCUS system and the CoCo among others, the* INPUT *statement will not allow a null string input. In this program I used the letter* **X** *as input to be tested and changed into the null string.*

And **INPUT** *seemed so trivial!]*

FLOPPY DISKS SALE *$1.19 ea.

Economy Model or Cadillac Quality

LORAN CERTIFIED PERSONAL COMPUTER DISK **_We have the lowest prices!_** LORAN CERTIFIED PERSONAL COMPUTER DISK

***ECONOMY DISKS**

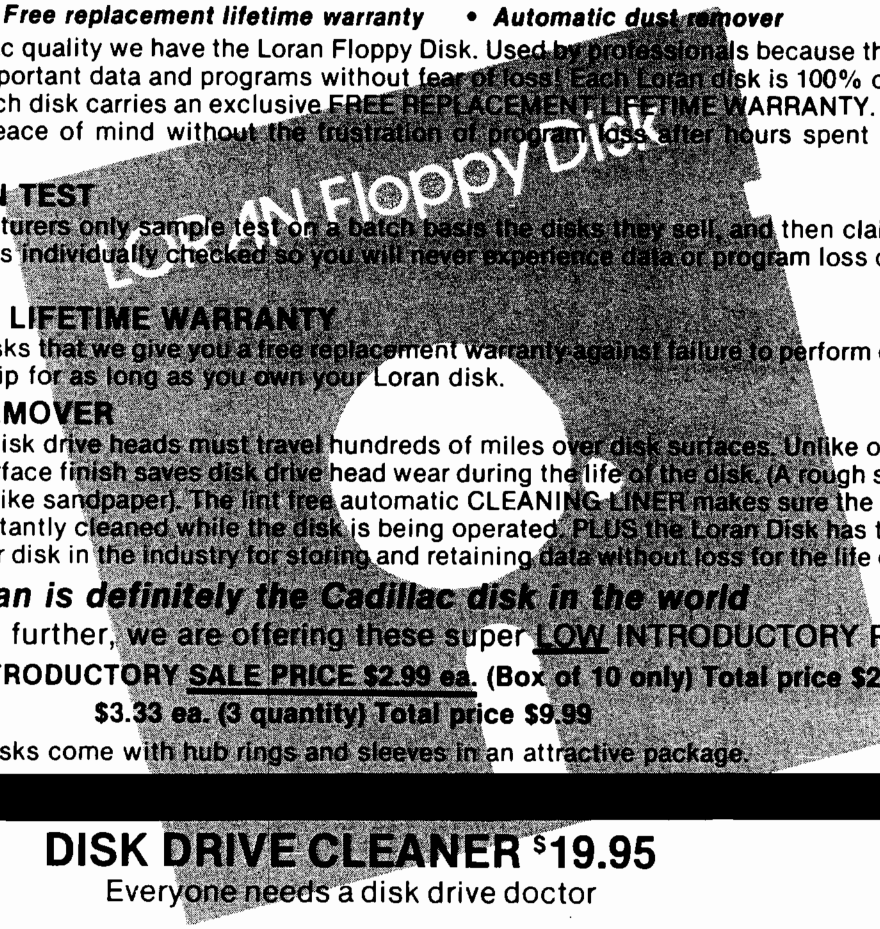
Good quality 5¼" single sided single density with hub rings.

Bulk Pac	100 Qty.	$1.19 ea.	Total Price	$119.00
	10 Qty.	1.39 ea.	Total Price	13.90

CADILLAC QUALITY

- ***Each disk certified*** • ***Free replacement lifetime warranty*** • ***Automatic dust remover***

For those who want cadillac quality we have the Loran Floppy Disk. Used by professionals because they can rely on Loran Disks to store important data and programs without fear of loss! Each Loran disk is 100% certified (an exclusive process) plus each disk carries an exclusive FREE REPLACEMENT LIFETIME WARRANTY. With Loran disks you can have the peace of mind without the frustration of program loss after hours spent in program development.



100% CERTIFICATION TEST

Some floppy disk manufacturers only sample test on a batch basis the disks they sell, and then claim they are certified. Each Loran disk is individually checked so you will never experience data or program loss during your lifetime!

FREE REPLACEMENT LIFETIME WARRANTY

We are so sure of Loran Disks that we give you a free replacement warranty against failure to perform due to faulty materials or workmanship for as long as you own your Loran disk.

AUTOMATIC DUST REMOVER

Just like a record needle, disk drive heads must travel hundreds of miles over disk surfaces. Unlike other floppy disks the Loran smooth surface finish saves disk drive head wear during the life of the disk. (A rough surface will grind your disk drive head like sandpaper). The lint free automatic CLEANING LINER makes sure the disk-killers (dust & dirt) are being constantly cleaned while the disk is being operated. PLUS the Loran Disk has the highest probability rate of any other disk in the industry for storing and retaining data without loss for the life of the disk.

Loran is definitely the Cadillac disk in the world

Just to prove it even further, we are offering these super LOW INTRODUCTORY PRICES

List $4.99 ea. INTRODUCTORY SALE PRICE $2.99 ea. (Box of 10 only) Total price $29.90

$3.33 ea. (3 quantity) Total price $9.99

All disks come with hub rings and sleeves in an attractive package.

DISK DRIVE CLEANER $19.95

Everyone needs a disk drive doctor

FACTS

- 60% of all drive downtime is directly related to poorly maintained drives.
- Drives should be cleaned each week regardless of use.
- Drives are sensitive to smoke, dust and all micro particles.
- Systematic operator performed maintenance is the best way of ensuring error free use of your computer system.

The Cheetah disk drive cleaner can be used with single or double sided 5¼" disk drives. The Cheetah is an easy to use fast method of maintaining efficient floppy diskette drive operation.

The Cheetah cleaner comes with 2 disks and is packed in a protective plastic folder to prevent contamination. List $29.95 / Sale $19.95

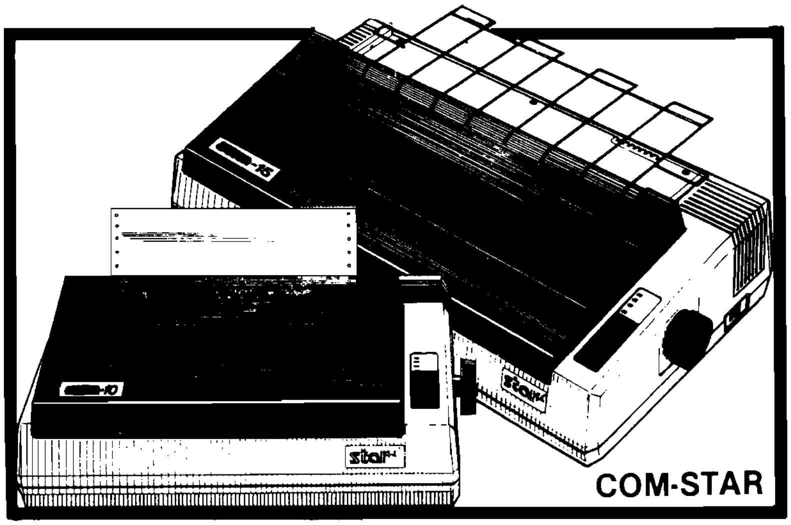
Add $10.00 for shipping, handling and insurance. Illinois residents please add 6% tax. Add $20.00 for CANADA, PUERTO RICO, HAWAII orders. WE DO NOT EXPORT TO OTHER COUNTRIES.

Enclose Cashiers Check, Money Order or Personal Check. Allow 14 days for delivery, 2 to 7 days for phone orders, 1 day express mail! Canada orders must be in U.S. dollars. Visa · MasterCard · C.O.D.

PROTECTO ENTERPRIZES (WE LOVE OUR CUSTOMERS)

BOX 550, BARRINGTON, ILLINOIS 60010

Phone 312/382-5244 to order



FANTASTIC PRINTER SALE

as low as **$149.00**

• *15 Day Free Trial - 180 Day Immediate Replacement Warranty*

	LIST	SALE
80 COLUMN THERMAL PRINTER — 60 CPS Bi-directional, dot matrix, prints 8½″ letter size paper, full 80 columns, high resolution graphics, dot bit addressable, special symbols and true decenders! (Centronics parallel interface)	$199	**$149**
80 COLUMN TRACTOR-FRICTION PRINTER — 80 CPS Bi-directional, dot matrix, impact, prints single sheets, continuous feed paper, adjustable columns, 40 to 132 characters! Roll paper adapter $32.95. (Serial or Centronics parallel interface)	$399	**$209**
PREMIUM QUALITY 10″ CARRIAGE T/F PRINTER — 120 CPS Bi-directional, impact, 9 x 9 dot matrix with double strike for 18 x 18 dot matrix. High resolution bit image (120 x 144 dot matrix) underlining back spacing, left and right margin settings, true lower decenders, with super and sub scripts. Prints standard, italic, block graphics, special characters, plus 24 of user definable characters and much more!! Prints single sheets, continuous feed and roll paper! (Centronics parallel interface)	$499	**$289**
PREMIUM QUALITY 15½″ CARRIAGE PRINTER — 120 CPS Has all the features of the Premium Quality 10″ Carriage T/F Printer above plus a 15½″ carriage and more powerful electronic components to handle large business forms! (Centronics parallel interface)	$599	**$379**
HIGH SPEED PREMIUM QUALITY T/F 10″ PRINTER — 160 CPS Save printing time with these plus features: 160 CPS speed, 100% duty cycle, 8K buffer diverse character fonts special symbols and true decenders, vertical and horizontal tabs. This is Red Hot Efficiency!!! (Serial or Centronics parallel interface)	$699	**$499**
HIGH SPEED PREMIUM QUALITY T/F 15½″ PRINTER — 160 CPS Has all the features of the 10″ Carriage high speed printer plus a 15½″ carriage and more powerful electronics to handle larger business forms! (Serial or Centronics parallel interface)	$799	**$599**

PARALLEL PRINTER INTERFACES: (IN STOCK)

- For VIC-20 and COMMODORE 64 — **$49.00**
- For all APPLE COMPUTERS — **$69.00**
- For ATARI 400 and 800 COMPUTERS — **$79.00**

NOTE: *Other printer interfaces are available at computer stores!*

SALE SALE

WE DO NOT EXPORT TO OTHER COUNTRIES EXCEPT CANADA.

Enclose Cashiers Check, Money Order or Personal Check. Allow 14 days for delivery, 2 to 7 days for phone orders, 1 day express mail! Canada orders must be in U.S. dollars. We accept Visa and MasterCard. We ship C.O.D.

PROTECTO
ENTERPRIZES (WE LOVE OUR CUSTOMERS)
BOX 550, BARRINGTON, ILLINOIS 60010
Phone 312/382-5244 to order

Announcing

The Winners of the **Graphics Contest**

Grand Prize

Title: **Taking the Bait**
System: **Commodore 64**
Graphics Package: **Koala Pad**
Computer Artiste:
Thaworn Phatinawin
1425 E. Ocean Blvd. #11
Long Beach, CA 90802

(This photo is on this month's cover. Another entry by Mr. Phatinawin titled: MICRO appeared on last month's MICRO cover.)

Congratulations from the staff of MICRO to all the winners of the Graphics Contest which was announced in the September, 1983 issue. The subjects and methods of presentation were all interesting, colorful, and varied. A surprising number included animation as well.
We truly enjoyed the efforts of everyone who entered, and we thank you for taking the time to participate.

Apple Winners

First Prize

Title: **Country 5**
System: **Apple II**
Graphics Package: **Koala/Micro Illustrator**
Computer Artiste:
Thomas Wilson
5 Cresta Circle #9
San Rafael, CA 94903

Second Prize
Title: **Polly Want a Cracker?**
by: **Lori Karoub**
Ypsilanti, MI 48197

Color Computer Winner

First Prize

Title: **Space Shuttle**
System: **Color Computer**
Graphics Package: **Personal Software**
Computer Artiste:
Eric White
375 Palm Springs Drive #1112
Altamonte Springs, FL 32701

Atari Winners

First Prize
Title: **Dragon Slayer**
System: **Atari 800**
Graphics Package: **Micropainter**
Computer Artiste:
Vic Albino
18501 194 NE
Woodinville, WA 98072

Second Prize
Title: **Loon Haven**
by: **Edward H. Cheely**
Accord, NY 12404

Third Prize
Title: **Landscape**
by: **Jane Zinke**
San Diego, CA 92115

Title: **Starwars Collection**
by: **Jim Stevenson**
Fairfax, VA 22033

Commodore Winner

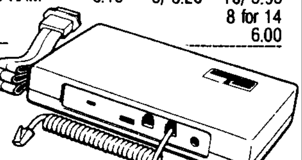
There was only **one** *entry in the Commodore class. This entry was so good that it won the* **Grand Prize**. *Too bad* **you** *did not take the time to enter – you would have* **WON**.

Hewlett Packard

Write or call for prices.

SPECIALS on INTREGATED CIRCUITS

6502	7.45	10/ 6.95	50/ 6.55	100/ 6.15
6520 PIA	5.15	10/ 4.90	50/ 4.45	100/ 4.15
6522 VIA	6.45	10/ 6.10	50/ 5.75	100/ 5.45
2716 EPROM	4.45	5/ 4.20	10/ 3.95	
2532 EPROM	5.90	5/ 5.75	10/ 5.45	
6116 2KX8 CMOS RAM	6.45	5/ 6.20	10/ 5.95	
4116 RAM			8 for 14	
4164 RAM			6.00	



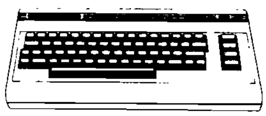
Anchor Automation Signalman Modems

FREE SOURCE MEMBERSHIP WITH SIGNALMAN

All Signalman Modems are Direct Connect, and provide the best price-performance values.

Dealer and OEM inquiries invited

Volksmodem with computer cable	(99)	68
Mark VI for IBM Personal Computer	(279)	139
Mark VII Auto Dial/Auto Answer	(159)	99
Mark XII Smart Model 1200/300	(399)	299
DC HAYES Smartmodem		219
DC Hayes Smartmodem 1200/300		529



PROM QUEEN for VIC	170
Apple Emulator for Commodore 64	Call
STAT Statistics Package for C64	95
Solid Oak 2 Level Stand for C64 or VIC	29
C64/VIC Switch (networking)	129
BACKUP V1.0 tape copier for C64 or VIC	20
CARDBOARD/6 Motherboard - VIC	64
CARDBOARD/5 Motherboard - C64	56
CARD PRINT G Printer Int. with Graphics	72
CARD PRINT B Printer Interface—C64/VIC	40
CARDBOARD/3s Motherboard - VIC	22
CARDCO C64/VIC Calculator Keypad	32
CARDRAM/16 RAM Expansion - VIC	44
Complete CARDCO Line in stock	
CIE and VIE IEEE Interfaces in stock	
MSD SuperDrive for C64 or IEEE	365
MAE Assembler for C64	50
Koala Touch Tablet—C64 or VIC	79
CBC 4/12 12-bit A to D Board + Software	199
Multiplan for C64	79
Dust Cover for C64 of VIC	6

APPLE—FRANKLIN ITEMS

FRANKLIN—complete line in stock	
QUENTIN Drives for Apple/Franklin	219
Swapper Stopper	26
automatic switch between paddles and joystick	
KRAFT Apple Joystick	40
Kraft Apple Paddle Pair	30
Koala Touch Tablet-Apple/Franklin	90
SPINNAKER Software in stock	
Broderbund Software in stock	
16K RAM Card for Apple	59
Multiplan—Microsoft	185
Solid Oak 2 Level Stand for Apple	29
Serial Card for Apple	89
MCP RAM/80 column card for IIe (AP/TXT)	139
Z80 Softcard and CP/M (Microsoft)	235
RANA Elite I with Controller	389
Parallel Printer Interface/Cable	69
Microtek and MCP Interfaces in stock	
Grappler + Interface	129
Kraft Products for Apple in stock	
DC Hayes Micromodem II	299
PFS: File	95
PFS: Report	95
Videx 80 Column Card	209
Apple Blue Book	19

commodore

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

PETSCAN I — $245 base price

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

COMPACK/STCP — $115

Intelligent Terminal Package for PET, CBM, C64

Includes ACIA Hardware / STCP Software

SCREENMAKER 80 Column Adapter for C64 — 145

Provides big screen capability for business applications.

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

VIC 20 Products and Software in stock

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

VICTORY Software for VIC and C64

Metamorphosis	16	Creator's Revenge	16
Labyrinth of Creator	16	Galactic Conquest	16
Kongo Kong	16	Annihilator	16
Chomper Man	16	Grave Robbers	13
Bounty Hunter	16	Adventure Pack I or II	16

PAPER CLIP Word Processor	99
ORACLE Data Base from Batteries Included	109
Super BusCard from Batteries Included	159
Commodore 64 Programmers Reference Guide	16
MicroChess for C64—8 levels of play	19
excellent graphics and color	
SPINNAKER Software C64, Apple, IBM, Atari	
Compute!'s First Book of PET/CBM	11
C64 or VIC SWITCH	129
POWER ROM Utilities for PET/CBM	78
WordPro 3+/64 with Spellmaster	85
WordPro 4+ - 8032, disk, printer	295
SPELLMASTER spelling checker for WordPro	
VISICALC for PET, ATARI, or Apple	189
PET-TRAX PET to Epson Graphics Software	40
SM-KIT enhanced PET/CBM ROM Utilities	40
Programmers Toolkit - PET ROM Utilities	35
EASY CALC for C64	65
PET Spacemaker II ROM Switch	36
COPYWRITER Word Processor for C64	69
2 Meter PET to IEEE or IEEE to IEEE Cable	40
Dust Cover for PET, CBM, 4040, or 8050	8
CmC Interfaces (ADA1800, ADA1450, SADI in stock)	
Programming the PET/CBM (Compute!) — R. West	20
Compute!'s Machine Language for Beginners	11
MULTIPLAN for C64	79
HES MODEM with Software	65
HES Software and Hardware in stock	
UMI products in stock	

FlexFile for PET/CBM/ C64 — $ 50

Database, Report Writer with calculations, Mailing Lists.

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

DISK SPECIALS

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

We stock VERBATIM DISKS

Write for Dealer and OEM prices.

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

We stock Dysan disks

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

We stock MAXELL DISKS

Write for dealer and OEM prices.

Disk Storage Pages 10 for $5 Hub Rings 50 for $6
Disk Library Cases 8"—3.00 5"—2.25
Head Cleaning Kits 12
AMARAY Disk Storage Systems in stock.
Innovative Concepts FLIP 'N' FILES in stock.

CASSETTE TAPES—AGFA PE-611 PREMIUM

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

DATASHIELD BACKUP POWER SOURCE — $265

Battery back up Uninterruptible Power Supply with surge and noise filtering. The answer to your power problems.

MultiPlan—IBM or Apple	185
Quadboard for IBM available	
KOALA Touch Tablets—Apple, Atari, IBM, CBM	
Peachtext 5000 Software Package	209
PFS Software for IBM and Apple in stock	
VOTRAX Personal Speech System	280
BMC 9191 Color Monitor	229
BMC 12A 12" Green Monitor	79
Brother HR-75 Daisy Wheel Printer	769
Dynax (Brother) DX-15 Daisy Wheel Printer	469
Itoh Prowriter Parallel Printer	379
Panasonic 1090 Printer with Correspondence Mode	279
Daisywriter 2000 with 48K buffer	1020
Gemini 10X	299
EPSON, Okidata, Star Micronics printers in stock	
USI CompuMOD 4 R F Modulator	29
We Stock AMDEK Monitors	
Amdek DXY-100 Plotter	590
A P Products	15% OFF
COMPUTER COVERUPS IN STOCK	
BROOKS 6 Outlet Surge Suppressor/Noise Filter	54
Surge Suppressor-6 outlet	29
Electrohome 1302-2 13" Hi-res RGB Monitor	335
Panasonic 12" Monitor (20 MHz) with audio	135
Synertek SYM-1 Microcomputer	189

ALL BOOK and SOFTWARE PRICES DISCOUNTED

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

ZENITH data systems

ZVM-122A	109	ZVM-123G	95
ZVM-131	300	ZVM-135	510
HERO 1 Robot (factory assembled)			2145
Z29 Terminal (DEC and ADM compatible)			680
ZT-10 Intel. Terminal with Serial Port			340
Z100 16-bit/8-bit Systems			CALL
We stock entire Zenith line.			

ATARI — WE STOCK ENTIRE LINE

SPINNAKER and Broderbund Software in Stock.

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

A B Computers

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

Credit Register

by Joseph Kattan

Keep track of credit card purchases as they occur to avoid overspending, and to be sure your monthly bills are correct

Requirements:
Any Atari Home Computer with minimum 24K RAM
Disk Drive

Most of us lose track some times of how much money we have spent on credit cards. Credit card companies do not supply us with registers for keeping track of such things and most of us just toss the little slips of paper that we receive after making a purchase with a credit card into some drawer where they can be safely forgotten. Sure, we have some idea of how much we have outstanding on credit cards, but then again every once in a while a bill comes in with some forgotten purchase, like that new half size disk drive. Credit Register will do away with all of these problems. The program will keep track of all of your credit card purchases on up to eight separate accounts and automatically update balances, billing information, and unbilled purchases.

Credit Register is a disk-based program, although it can be easily modified to run on a 16K tape system. The first menu in the program presents you with three choices: Review Files, Revise Files, and Create Files. When you run the program for the first time, choose the "Create Files" option. This will write to a data disk a grid of arrays matrices in which your data can be stored. After that initial use, you can create new accounts, revise account data, and delete accounts using the "Revise Files" option.

If you only wish to review the status of any account, choose the "Review Files" menu option. To enter data into the program, choose the "Revise Files" option. You will find this to be the most commonly used entry on the main menu. When you choose either option, Credit Register will prompt you to enter your data disk into the disk drive. Your may, of course, store your data on the same disk as the program, in which case you will not have to swap disks. Once you have the proper disk in the drive and have pressed {RETURN}, the program will load your data files into memory and display a list of eight accounts. If you have not named any account, it will appear on the screen display as "Unused." You will then be asked to enter the number (from 1 to 8) of the account which you wish to review or revise. If you are creating a new account, enter a number of an "Unused" account. Let us assume here that you have chosen an account for which data has been previously entered. The screen display should look like Figure 1.

```
ACCOUNT: VISA              ORIG BAL: 0.00
NEW AMOUNT BILLED: 307.57
BILL TOTAL: 307.57          DUE: 11/20/83

01    10/07/83 DISCS          45.00
02    10/14/83 AMTRAK         53.00
03    10/31/83 TOYS           21.89
04    10/31/83 BOOKS          16.17
05    10/31/83 COSTUME        23.73
06    11/11/83 PAC MAN        34.78
07    11/11/83 FABRIC         42.76
08    11/12/83 BOOKS          15.74

TOTAL: 560.64           UNBILLED: 253.07

1) PURCHASE  2) BILLING  3) PAYMENT
4) EDIT ACCT  5) ACCT NAME  6) EXIT
```

Menu entry 1 allows you to enter new purchase information, entry 2 billing information, and entry 3 payment information. If you choose any of these options, the program will lead you step-by-step through the proper data entry procedures. It will prompt you to enter the date in a month/day/year format over a display, enter the amount of the purchase or bill, and a name for a purchase. When you enter billing information, the program will ask you to identify the purchases that have been billed and will place a "B" next to any billed item until you have completed entering the billing information. When you have done so, all billed items will be deleted from the list of unbilled purchases and the screen display undated to reflect that revision. Menu item 5 should be used only to establish or revise an account name. Menu item 6 allows you to exit the account revision mode.

Menu item 4, "Edit Acct," allows you to correct erroneous data in the account's file. When you choose that option, you will be presented with a new menu:

```
1) ORIG BAL  2) LAST BILL  3) EXIT
4) DELETE PURCHASE  5) DELETE ACCT
```

You may modify the original balance, the amount of the last bill, delete any purchase, or delete the entire account by entering the appropriate item number on this menu. This part of the program is self-explanatory and the program will lead you through the correct entry procedures step-by-step.

Credit Register comes with restrictions and will inform you if you violate any of them. The program will not accept more than 12 unbilled purchases for any account; it will tell you that if you attempt to exceed the limit. The program also limits you to an eight-character name for any account or purchase. Again, if you attempt to exceed it, Credit Register will remind you of the limit and give you another opportunity to enter the data. If a bill amount that you entered does not match the sum of the purchases that you had told the program were on the bill, Credit Register will give you an opportunity to reconcile the discrepancy. If you attempt to register a payment when there is no bill outstanding, the program will inform you of the problem. The program will also refuse to accept an invalid date. In general, Credit Register will not accept incorrect entries or will advise you of any apparent problems in data that you are entering.

Every account display in the program will show all unbilled purchases, the original account balance, the amount of new items reflected on the last bill, the total amount of the last bill, the total amount outstanding on the account, and the total value of unbilled purchases. When you are revising accounts, the program will revise the total in each category and identify purchases that have been billed or deleted until you complete the data entry process, at which time a new screen will be drawn without the billed or deleted items.

A note on data entry. At any point in the program at which you are allowed to press any key, you may return to the main Credit Register menu by pressing the {OPTION} key. Thus, if you wander into any part of the program by mistake, you can always leave it by pressing {OPTION}. The only drawback to this method of escape is that program memory is cleared of all data that you might have entered.

The data entry routine allows this method of escape because the INPUT statement is never used in Credit Register. Instead, the subroutine at lines 300-312 accepts individual keystrokes, tests them, and accepts them only if they are valid. For that reason, the cursor control keys (arrow keys) are ignored by the program. If Credit Register expects numerical data, it will ignore all keys except the numbers 0 through 9, the period mark and, of course, the {OPTION} key. The date entry routine beginning at line 240 similarly tests individual keystrokes and accepts only numerical entries and the {OPTION} key. Both routines were used in my program, The Investor, in the February 1984 MICRO and are explained in that article.

The routines at lines 680 through 760 are used to transfer memory image files from disk to RAM and vice versa using the Atari's resident disk handler. Credit Register uses this technique to save and load the data files that you create using the program. The advantage of this technique over using INPUT # and PRINT # statements is speed. The program loads the entire data file, consisting of over 1000 string array elements and a 9 by 14 matrix of floating point numbers in about one second! The technique is similar to those used to save and load special fonts and graphic screens.The starting address in RAM of the data to be saved or loaded and the length of the area of memory to be saved or loaded are POKEd into the appropriate addresses in a special buffer used by the Atari for input/output operations, together with a read or write command. The memory locations and commands are listed in Ian Chadwick's *Mapping the Atari* at pp. 83-89. The USR command then passes control of the program to the resident disk handles, which handles the data transfer in a jiffy.

Because of the use of this special method of communicating with the disk drive, it is absolutely imperative that the DIM statement in line 10 be typed in exactly as it appears in the program listing. Atari BASIC allocates RAM to strings and arrays in the order in which it encounters them. Since we are POKEing into the I/O buffer an address of the area in RAM to be affected by the transfer of data to and from the disk drive, the BASIC interpreter must encounter our strings, arrays, and matrix in the order in which they appear in line 10. Otherwise, the data we wish to transfer will reside in a different area of memory than that which the disk handler is told to transfer.

The program's main data is stored in three strings and a matrix. ACT$ keeps track of all account names, BILL$ stores billing dates, and DATE$ keeps track of the dates and names of all purchases. The amounts of purchases are stored in the AMT matrix. Note that both BILL$ and DATE$ concatenate each date to three bytes using the CHR$ function. Since any month or day has a value of less than 255 and thus may be stored in the form of a single character string, the month and day are converted to a character string. The year is converted to a character string after subtracting 1900 from its value. Thus, the year 1983 is stored as a CHR$(83).

The program has a few interesting bells and whistles that should be mentioned. The short subroutine at lines 800 through 810 is used for decimal justification of numerical data. The subroutine changes all numbers into a dollars and cents format by adding the trailing zeros to integers and multiples of 0.1. The program also makes use of the LOCATE statement, which I have not encountered before in a non-graphic use. It uses the statement to keep track of accounts that have been deleted or billed until all modifications are completed and thereby prevents a second deletion or billing of the same item. It keeps track of the status of the item by looking for a B or D character next to the dollar value of the item. Finally, the routine at lines 960 through 970 avoids a complex sorting algorithm to get rid of billed or deleted purchases by wasting a little memory and transferring the contents of one array into another array and then back into the original array, instead of passing values from the array into a single variable and vice versa. It is just a reminder that we can sometimes speed program execution by wasting a little memory, something we can do with this program.

Listing 1

```
5 OPEN #5,4,0,"K:"
10 DIM DATE$(1056),AMT(8,13),BILL$(24),ACT$(64),Q$(13),A$(8),
   BL$(20),S(8),DISC$(1),SUB$(132),SUB(12)
20 DATE$=" ":DATE$(1056)=DATE$:DATE$(2)=DATE$:BL$=DATE$(1,20):
   ACT$=DATE$(1,64):SUB$=DATE$(1,144):BILL$=DATE$(1,24)
30 FOR I=1 TO 8:FOR N1=0 TO 13:AMT(I,N1)=0:NEXT N1:NEXT I
50 GRAPHICS 0:POKE 710,160:POKE 712,146:POKE 752,1:ADT=ADR(DATE$)-1:
   POKE 16,64:POKE 53774,64
100 PRINT "{CLEAR}":POSITION 10,10:PRINT "<1> REVIEW FILES":POKE 85,10:
    PRINT "<2> REVISE FILES":POKE 85,10:PRINT "<3> CREATE FILES"
105 MX=3:GOSUB 155:GOTO N1*1000
155 POSITION 11,20:PRINT "ENTER NUMBER: ";
160 C=PEEK(85):R=PEEK(84):A$="NUM":GOSUB 300:
    IF N1<1 OR N1>MX THEN POSITION C,R:GOTO 160
165 RETURN
170 PRINT "CHANGE DATA";
175 PRINT " ({REVERSE Y}ES OR {REVERSE N}O)? ";
180 GOSUB 300:A$=Q$(1,1):IF A$="Y" OR A$="N" THEN PRINT "{UP,DELETE LINE}";:
    RETURN
195 POSITION C,R:GOTO 180
240 GOSUB 450:N=PEEK(85)-1:S=PEEK(84):
    REM DATE ENTRY ROUTINE-ENTER DAY,MONTH,YEAR
241 TRAP 241:POSITION N+1,S:PRINT "{CTRL E}-/--/----":J=1:A$=""
242 POSITION N+J+(J>2)+(J>4),S:PRINT "{CTRL E}":IF PEEK(53279)=3 THEN 312:
    REM ON OPTION KEY, CLEAR MEMORY AND GO TO MENU
243 IF PEEK(764)=255 THEN 242
244 GET #5,A:IF A=126 AND J<>1 THEN POSITION N+J+(J>2)+(J>4),S:
    PRINT "-":J=J-1:GOTO 242
245 IF A<48 OR A>57 THEN 242
246 POSITION N+J+(J>2)+(J>4),S:PRINT CHR$(A):S(J)=A:J=J+1:IF J<9 THEN 242
248 GET #5,A:IF A=126 THEN J=8:GOTO 242
250 IF A<>155 THEN 248
252 FOR J=1 TO 8:A$(J,J)=CHR$(S(J)):NEXT J:M=VAL(A$(1,2)):
    D=VAL(A$(3,4)):Y=VAL(A$(5,8))
253 IF M=0 OR M>12 OR D=0 OR D>31 OR (M=2 AND D>29) OR
    Y<1900 OR Y>2155 THEN 256
254 POSITION 2,22:PRINT "{DELETE LINE}":POSITION 2,S+1:IF Q$="BIL" THEN RETURN
255 POKE I-10,M:POKE I-9,D:POKE I-8,Y-1900:RETURN
256 POSITION 2,22:PRINT "{BEEP}{DELETE LINE}PLEASE REENTER CORRECT DATE":
    GOTO 241
295 IF N1<10 THEN PRINT "0";
296 RETURN
300 C=PEEK(85):R=PEEK(84)
301 TRAP 301:Q$="":POSITION C,R:PRINT BL$:Y=1:S=32
302 POKE 764,255:A=0:POSITION C+Y-1,R:PRINT "{CTRL E}";
303 IF PEEK(53279)=3 THEN 312
304 IF PEEK(764)=255 THEN 303
305 GET #5,A:IF A=155 AND (Y>1 OR A$<>"NUM") THEN 310
306 IF A=126 THEN Y=Y-(Y>1):Q$=Q$(1,Y):POSITION C+Y,R:PRINT " ":GOTO 302
307 IF A<32 OR A>124 OR (A$="NUM" AND (A<48 OR A>57) AND A<>46) THEN 302
308 Q$(Y)=CHR$(A):POSITION C+Y-1,R:PUT #6,A:IF Y=1 THEN S=A
309 Y=Y+1:GOTO 302
310 Q$(1,1)=CHR$(S):PRINT "{LEFT,DELETE}":IF A$="NUM" THEN N1=VAL(Q$):A$="    "
311 S=LEN(Q$):RETURN
312 POKE 710,160:CLR :GOTO 10
320 PRINT "ENTER ACCOUNT NAME: ";
325 GOSUB 300:IF S>8 THEN POKE 84,20:
    PRINT "{BEEP}ACCT NAME CANNOT EXCEED 8 CHARACTERS":POSITION C,R:GOTO 325
330 IF S<8 THEN Q$(S+1,8)=BL$(1,8-S)
332 IF A$="RE" THEN A$="":RETURN
333 ACT$(ID*8-7,ID*8)=Q$(1,8):PRINT
335 PRINT "ENTER STARTING BALANCE: ";:A$="NUM":GOSUB 300:AMT(ID,0)=N1:RETURN
```

```
350 IF COUNT=13 THEN PRINT "{BEEP}        SORRY, ACCOUNT FULL":GOSUB 390:RETURN
352 PRINT "DATE OF TRANSACTION: ";:GOSUB 240:PRINT "PURCHASE: ";:A$="RE":
    GOSUB 325:DATE$(AI*11-7,AI*11)=Q$(1,8)
355 POSITION 24,PEEK(84)-1:PRINT "AMT: ";:A$="NUM":GOSUB 300:AMT(ID,COUNT)=N1
360 COUNT=COUNT+1:RETURN
390 IF PEEK(84)<23 THEN PRINT
392 POKE 85,8:PRINT "PRESS {REVERSE RETURN} TO CONTINUE";
395 POKE 764,255
396 IF PEEK(53279)=3 THEN 312
397 IF PEEK(764)=255 THEN 396
398 GET #5,A:IF A<>155 THEN 396
399 RETURN
400 PRINT "{CLEAR}ACCT: ";ACT$(ID*8-7,ID*8);:
    IF ACT$(ID*8-7,ID*8)=BL$(1,8) THEN POKE 85,8 PRINT "UNUSED";
405 POKE 85,19:PRINT "ORIG BAL: ";:N1=AMT(ID,0):GOSUB 800:PRINT Q$
406 IF AMT(ID,13)=0 THEN 410
407 PRINT "NEW AMOUNT BILLED: ";:N1=AMT(ID,13):GOSUB 800:PRINT Q$
408 PRINT "BILL TOTAL: ";:N1=AMT(ID,0)+AMT(ID,13):GOSUB 800:PRINT Q$;:
    POKE 85,26:PRINT "DUE ";:N1=ASC(BILL$(ID*3-2,ID*3-2))
409 GOSUB 295:PRINT N1;"/";:N1=ASC(BILL$(ID*3-1,ID*3-1)):GOSUB 295:
    PRINT N1;"/";:N1=ASC(BILL$(ID*3,ID*3)):GOSUB 295:PRINT N1
410 AMTN=AMT(ID,0)+AMT(ID,13):PRINT :PL=PEEK(84)-1:FOR COUNT=1 TO 12:
    GOSUB 450:IF AMT(ID,COUNT)=0 THEN POP :GOTO 440
415 M=PEEK(I-10):D=PEEK(I-9):Y=PEEK(I-8):N1=COUNT:GOSUB 295:PRINT COUNT;"  ";
420 N1=M:GOSUB 295:PRINT M;"/";:N1=D:GOSUB 295:PRINT D;"/";:N1=Y:GOSUB 295:
    PRINT Y;" ";
430 PRINT DATE$(AI*11-7,AI*11);"   ";:N1=AMT(ID,COUNT):GOSUB 800:
    POKE 85,35-S:PRINT Q$:AMTN=AMTN+AMT(ID,COUNT):NEXT COUNT
440 PRINT :PRINT "TOTAL: ";:N1=AMTN:GOSUB 800:PRINT Q$;:
    PRINT " UNBILLED: ";:N1=AMTN-AMT(ID,0)-AMT(ID,13):GOSUB 800:
    PRINT Q$:RETURN
450 AI=ID*12-12+COUNT:I=AI*11+ADT:RETURN
500 PRINT "{CLEAR}":POKE 85,12:PRINT "ACCOUNTS IN FILE":PRINT :PRINT :
    FOR I=1 TO 8:S=(I/2=INT(I/2)):POKE 85,2+S*18:N1=I:GOSUB 295
510 PRINT I;:POKE 85,8+S*18:IF ACT$(I*8-7,I*8)=BL$(1,8) THEN PRINT
    "UNUSED";:GOTO 530
520 PRINT ACT$(I*8-7,I*8);
530 IF S THEN PRINT
540 NEXT I:RETURN
550 IF COUNT=1 THEN PRINT "{BEEP}NOTHING TO BILL IN DATA BASE. UPDATE
    BILLING";:GOSUB 175:PRINT :IF A$="N" THEN RETURN
551 IF A$="SKIP" THEN 557
552 IF AMT(ID,13)<>0 THEN PRINT "{BEEP} PAYMENT ON LAST BILL NOT ENTERED":
    GOSUB 390:RETURN
553 PRINT "NEW PURCHASES BILLED: $";:A$="NUM":GOSUB 300:AMT(ID,13)=N1:
    IF AMT(ID,0)=0 THEN INT=0:GOTO 555
554 PRINT "INTEREST BILLED: $";:A$="NUM":GOSUB 300:INT=N1:AMTZ=AMTZ+INT:
    AMT(ID,13)=AMT(ID,13)+INT
555 PRINT "BILL PAYMENT DATE: ";:Q$="BIL":GOSUB 240:
    BILL$(ID*3-2,ID*3-2)=CHR$(M):BILL$(ID*3-1,ID*3-1)=CHR$(D)
556 BILL$(ID*3,ID*3)=CHR$(Y-1900)
557 IF COUNT=1 THEN 575
558 PRINT "ITEMS BILLED (1 TO ";COUNT-1;", ";COUNT;" IF NONE): ";
559 A$="NUM":GOSUB 300:IF N1<1 OR N1>COUNT THEN POSITION C,R:GOTO 559
560 IF N1=COUNT THEN 575
561 LOCATE 36,PL+N1,A:PRINT "{LEFT}";CHR$(A):IF A=66 THEN POSITION C,R:GOTO 559
562 AMTZ=AMTZ+AMT(ID,N1):AMT(ID,N1)=0:POSITION 36,PL+N1:PRINT "B":POKE 84,R+1
570 PRINT "MORE ITEMS BILLED";:GOSUB 175:IF A$="Y" THEN PRINT
    "{UP,DELETE LINE}";:GOTO 558
575 GOSUB 950
580 IF AMTZ=AMT(ID,13) THEN 610
585 GOSUB 400:PRINT :PRINT "{BEEP}NEW PURCHASES BILLED: ";:N1=AMT(ID,13)-INT:
```

```
    GOSUB 800:PRINT Q$:PRINT "BUT BILLED ITEMS TOTAL: ";
590 N1=AMTZ-INT:GOSUB 800:PRINT Q$:PRINT "INTEREST BILLED: ";:N1=INT:GOSUB 800:
    PRINT Q$:PRINT "CHANGE AMOUNT BILLED";
595 GOSUB 175:IF A$="N" THEN 605
600 PRINT "NEW PURCHASES BILLED: $";:A$="NUM":GOSUB 300:AMT(ID,13)=N1:
    AMT(ID,13)=AMT(ID,13)+INT:GOTO 580
605 PRINT "START OVER";:GOSUB 175:IF A$="Y" THEN CLR :GOTO 10
610 RETURN
625 POP :RETURN
650 PRINT "ENTER AMOUNT PAID: ";:A$="NUM":GOSUB 300:
    AMT(ID,0)=AMT(ID,0)-N1+AMT(ID,13):AMT(ID,13)=0:RETURN
680 IF DISC$="Y" THEN RETURN
685 TRAP 695:GOSUB 700:OPEN #1,4,0,"D:CREDIT.DAT":GOSUB 750:
    POKE 850,7:POKE 858,4:A=USR(ADR(Q$))
690 CLOSE #1:DISC$="Y":RETURN
695 PRINT "{BEEP}":POKE 712,64:FOR DELAY=1 TO 100:NEXT DELAY:POKE 712,146:
    CLOSE #1: GOTO 680
700 POKE 764,255:PRINT "{CLEAR}":POKE 84,11:
    PRINT " INSERT DATA DISC AND PRESS {REVERSE RETURN}": GOSUB 395:RETURN
710 PRINT :PRINT "WRITE CHANGES TO DISC";
715 GOSUB 175:IF A$="N" THEN RETURN
720 GOSUB 700:TRAP 760:OPEN #1,8,0,"D:CREDIT.DAT":GOSUB 750:
    POKE 850,11:POKE 858,8:A=USR(ADR(Q$))
735 CLOSE #1:DISC$="Y":PRINT "{CLEAR}":RETURN
750 Q$="h{REVERSE ",CTRL P}LV{REVERSE d}":S=INT(ADR(DATE$)/256):
    N1=ADR(DATE$)-256*S:MX=(ADR(Q$)-ADR(DATE$)):R=INT(MX/256):C=MX-R*256
755 POKE 852,N1:POKE 853,S:POKE 856,C:POKE 857,R:RETURN
760 PRINT "{UP,DELETE LINE}     PROBLEMS WITH DISC DRIVE":GOSUB 390:GOTO 720
800 N1=INT(N1*100+0.5)/100:Q$=STR$(N1):
    IF N1=INT(N1) THEN Q$(LEN(Q$)+1)=".00":GOTO 810
805 IF N1*10=INT(N1*10) THEN Q$(LEN(Q$)+1)="0"
810 S=LEN(Q$):RETURN
850 PRINT "1) ORIG BALANCE  2) LAST BILL  3) EXIT4) DELETE PUR
    CHASE 5) DELETE ACCT"
855 MX=5:GOSUB 160:PRINT "{UP,DELETE LINE}":ON N1 GOSUB 870,875,625,885,925
860 PRINT "MORE EDITING";:GOSUB 175:IF A$="Y" THEN GOSUB 400:PRINT :GOTO 850
865 RETURN
870 GOSUB 335:RETURN
875 PRINT "NEW PURCHASES ON BILL :$";:A$="NUM":GOSUB 300:AMT(ID,13)=N1
880 PRINT "INTEREST ON BILL: $";:A$="NUM":GOSUB 300:
    AMT(ID,13)=AMT(ID,13)+N1:RETURN
885 IF COUNT=1 THEN PRINT "{BEEP}NO PURCHASES TO DELETE IN DATA BASE":
    GOSUB 390:PRINT "{UP,DELETE LINE,UP,DELETE LINE}":GOTO 860
890 PRINT "NO. OF ITEM (1 TO ";COUNT-1;", ";COUNT;" IF NONE): ";
892 A$="NUM":GOSUB 300:IF N1<1 OR N1>COUNT THEN POSITION C,R:GOTO 892
893 IF N1=COUNT THEN 900
894 LOCATE 36,PL+N1,A:PRINT "{LEFT}";CHR$(A):IF A=68 THEN POSITION C,R:GOTO 892
896 AMTZ=AMTZ-AMT(ID,N1):AMT(ID,N1)=0:POSITION 36,PL+N1:PRINT "D":POKE 84,R+1
898 PRINT "MORE ITEMS TO DELETE ";:GOSUB 175:
    IF A$="Y" THEN PRINT "{UP,DELETE LINE}";:GOTO 890
900 GOSUB 950:RETURN
925 ACT$(ID*8-7,ID*8)=BL$(1,8):SUB$=" ":SUB$(132)=SUB$:SUB$(2)=SUB$:
    DATE$(ID*132-131,ID*132)=SUB$
930 FOR J=0 TO 13:AMT(ID,J)=0:NEXT J:RETURN
950 FOR J=1 TO 12:SUB(J)=0:NEXT J:SUB$=" ":SUB$(132)=SUB$:SUB$(2)=SUB$
960 J=1:FOR COUNT=1 TO 12:GOSUB 450:IF AMT(ID,COUNT)=0 THEN 970
965 SUB(J)=AMT(ID,COUNT):SUB$(J*11-10,J*11)=DATE$(AI*11-10,AI*11):J=J+1
970 NEXT COUNT:FOR COUNT=1 TO 12:AMT(ID,COUNT)=SUB(COUNT):NEXT COUNT:
    DATE$(ID*132-131,ID*132)=SUB$:RETURN
1000 GOSUB 680
1010 GOSUB 500:POKE 84,18:PRINT "NO. OF ACCT TO REVIEW (9 TO EXIT)";:MX=9:
     GOSUB 155:ID=N1:IF N1=9 THEN CLR :GOTO 10
```

```
1020 GOSUB 400:GOSUB 390:GOTO 100
2000 GOSUB 680
2010 GOSUB 500:POKE 84,18:PRINT "NUMBER OF ACCT TO CHANGE (9 TO EXIT)":MX=9:
     GOSUB 155:ID=N1:IF N1=9 THEN 50
2020 GOSUB 400:PRINT :PRINT "1) PURCHASE  2) BILLING  3) PAYMENT":
     PRINT "4) EDIT ACCT 5) ACCT NAME  6) EXIT"
2030 PRINT :PRINT "ENTER NUMBER: ";:MX=6:GOSUB 160:PRINT "{UP,DELETE LINE,
     UP,DELETE LINE,UP,DELETE LINE,UP,DELETE LINE,UP,DELETE LINE}":
     AMTZ=0:CHOICE=N1:ON N1 GOSUB 350,550,650,850,320,610
2035 IF CHOICE=6 THEN 2050
2040 PRINT "{UP,DELETE LINE,UP,DELETE LINE}":PRINT "MORE CHANGES ON ACCT";:
     GOSUB 175:IF A$="Y" THEN 2020
2050 GOSUB 400:GOSUB 710:IF A$="N" THEN GOSUB 390
2060 GOTO 100
3000 POKE 710,96:PRINT "{CLEAR}":POKE 84,10:PRINT "CAUTION: WRITING TO DISC
     WILL ERASE EXISTING DATA FILES ON THE DISC."
3010 PRINT :PRINT "   WRITE TO DISC";:GOSUB 175:POKE 710,160:IF A$="N" THEN 100
3020 GOSUB 720:GOTO 100
```

Ed. Note: Microbes from the listings of Mr. Kattan's previous article, "The Investor" (Micro 69:19), appear on page 73.

Joseph Kattan is a lawyer in Washington. He may be written to at 5721 Chevy Chase Pkwy.N.W., Washington, DC 20015

MICRO™

CoCo Bits

by John Steiner

OS-9

OS-9 is becoming more and more popular among CoCo hackers. I have gotten a few letters from readers about their experiences with OS-9. Gene Driskell of Xenia, OH called and wrote about a problem with OS-9 and the external terminal mode. It seems the CoCo would occasionally garble a letter being sent from the terminal to the CoCo. I wasn't much help with the problem as I had not run into it, and no one else I talked with had either. A few days later, Gene sent me another letter. As it turns out, OS-9 will only support terminal I/O at 300 baud. Mr. Driskell had been using the default terminal baud rate of 600. At 600 baud, Gene said that only about 13% of the characters were incorrectly sent.

It would be nice to operate the terminal at a higher baud rate; if you have a patch, or know of a way around this problem, pass the information along and I will relay it here. I will have some comments on BASIC-09 in a few moments.

CoCo 2 Memory Expansion

Santa has brought a lot of Color Computer 2's into people's homes, and the small powerhouse has introduced many to the joys of computing. With 16K the stock memory, many people are opting to raise the memory requirements to 64K. The process is easy.

There are 2118s in the CoCo 2, rather than 4116s as in the earlier CoCos. To upgrade to 64K, all it takes is replacement of the chips and soldering a single jumper The jumper is located near the 6822 and 74LS244 chips. A lettering on the PC board reads W1. Immediately adjacent to the label, W1 and toward the rear of the board from the label are two solder pads. These two pads should be jumpered together. Thanks to Gene for providing the upgrade instructions.

By the way, the piggyback 32K upgrade used before the 64K machines were introduced is still possible. The 2118s require only a +5 volt line, rather than the +5, -5 and +12 volts required of the 4116s. As a result, you cannot piggyback 4116s with the 2118s.

Software Speech Synthesizer

Classical Computing, Inc. of Chapel Hill, NC sent me a copy of SPEAK UP, a machine language voice synthesizer for the CoCo. The program, written by David Dubowski, is an excellent example of software voice synthesis. I have had a lot of fun with its ability to speak any standard BASIC string. It also has the ability to read the screen and speak any phrase printed there. The computer will speak the words until it finds a period, question mark or exclamation point. The price you pay for making your programs talk is memory. The program requires just over 7K of RAM. Phonomes are used to generate the parts of speech, so you have to misspell some words in order to make them sound right as the computer speaks them. For example, CHAMPAGNE sounds best when spelled SHAMPAYN. At only $29.95, its quite a bargain.

More CoCo Rumors

I have been hearing from the grape vine about a new CoCo to be released shortly. Talk is of a 256K CoCo. This word comes just after the release of the Tandy 2000, an IBM compatible computer with sophisticated color capacity. My first thought was that the 2000, since it is 256K, was what they were referring to. Even now that the 2000 has been released, the rumors persist. Tandy kept the 2000 a big secret until just before its release, so any new CoCos will probably be just as big a secret. We shall see.

BASIC-09

With the arrival of OS-9, a new world of BASIC programming is at the CoCo keyboard. BASIC-09 is one of the most powerful versions of BASIC I have seen. I have had access to many versions recently, as I have just finished a manuscript for Prentice-Hall that will be released sometime this year. The book is a BASIC cross-referencing dictionary. In my research, one BASIC caught my eye as being especially powerful, BASIC-09. At the time, I hadn't really expected it to be released for CoCo. Microware provided me a BASIC manual for use with my research. My CoCo version is still on order as I write this, but I am assuming there will be little difference between Radio Shack's version and the standard Microware release.

Probably the most unique feature of BASIC-09 is its Pascal-like procedures. Line numbers are optional within procedures, and several procedures may reside in memory at any given time. It is possible to load and save multiple procedures in one step, and any procedure currently in memory may be used by either the operator of the main keyboard, or by any terminal user. Procedures may be called from within other procedures, and each may be tested and debugged individually, if desired.

Those familiar with Color BASIC, or any BASIC for that matter, recognize the FOR-NEXT loop. BASIC-09 uses several loop structures in addition to FOR-NEXT. Here are a few examples.

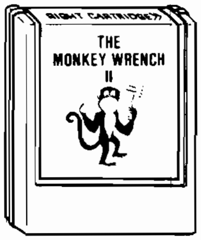
Products for Commodore, Atari, Apple, and others!

NEW

THE MONKEY WRENCH II

A PROGRAMMERS AID FOR ATARI 800
NEW AND IMPROVED — 18 COMMANDS
PLUGS INTO RIGHT CARTRIDGE SLOT

If you are a person who likes to monkey around with the ATARI 800, then THE MONKEY WRENCH II is for you!! Make your programming tasks easier, less time-consuming and more fun. Why spend extra hours working on a BASIC program when the MONKEY WRENCH can do it for you in seconds. It can also make backup copies of boot type cassette programs. Plugs into the right slot and works with ATARI BASIC cartridge.



$59.95

The MONKEY WRENCH provides 18 direct mode commands. They are: AUTO LINE NUMBERING — Provides new line numbers when entering BASIC program lines. RENUMBER — Renumbers BASIC's line numbers including internal references. DELETE LINE NUMBERS — Removes a range BASIC line numbers.

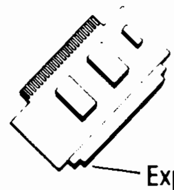
VARIABLES — Display all BASIC variables and their current value. Scrolling — Use the START & SELECT keys to display BASIC lines automatically. Scroll up or down BASIC program. FIND STRING — Find every occurrence of a string. XCHANGE STRING — Find every occurrence of a string and replace it with another string. MOVE LINES — Move lines from one part of program to another part of program. COPY LINES — Copy lines from one part of program to another part of program. FORMATTED LIST — Print BASIC program in special line format and automatic page numbering. DISK DIRECTORY — Display Disk Directory. CHANGE MARGINS — Provides the capability to easily change the screen margins. MEMORY TEST — Provides the capability to test RAM memory. CURSOR EXCHANGE — Allows usage of the cursor keys without holding down the CTRL key. UPPER CASE LOCK — Keeps the computer in the upper case character set. HEX CONVERSION — Converts a hexadecimal number to a decimal number. DECIMAL CONVERSION — Converts a decimal number to a hexadecimal number. MONITOR — Enter the machine language monitor.

In addition to the BASIC commands, the Monkey Wrench also contains a machine language monitor with 16 commands used to interact with the powerful features of the 6502 microprocessor.

NEW

VIC RABBIT CARTRIDGE AND CBM 64 RABBIT CARTRIDGE

"High-Speed Cassette Load and Save!"



$39.95
(includes Cartridge and Manual)

Expansion Connector on the VIC Cartridge

"Don't waste your Life away waiting to LOAD and SAVE programs on Cassete Deck."

Load or Save 8K in approximately 30 seconds! Try it — your Un-Rabbitized VIC takes almost 3 minutes. It's not only Fast but VERY RELIABLE.

Almost as fast as VIC Disk Drive! Don't be foolish — Why buy the disk when you can get the VIC Rabbit for much, much less!

Easy to install — it just plugs in.
Expansion Connector on rear.
Works with or without Expansion Memory.
Works with VIC Cassette Deck.
12 Commands provide other neat features.

Also Available for 2001, 4001, and 8032

TELSTAR 64

Sophisticated Terminal Communications Cartridge for the 64.

PFO IOD OOD CP<D1>D2 BELL 12:30:00 10:14:36
(TELSTAR's Status Line)

Don't settle for less than the best!

- Upload/Download to/from disk or tape.
- Automatic File Translation.
- Communicates in Industry Standard ASCII.
- Real-Time Clock plus Alarm Clock.
- Line editing capability allows correcting and resending long command lines.
- 9 Quick Read functions.
- Menu-driven.
- Similar to our famous STCP Terminal package.
- Works with Commodore Modems and supports auto-dialing.

The best feature is the price — **only $49.95** (Cartridge and Manual)

Machine Language Monitor Cartridge

for the CBM 64

More than 20 commands allow you to access the CBM 64's Microprocessors Registers and Memory Contents. Commands include assemble, disassemble, registers, memory, transfer, compare, plus many more.

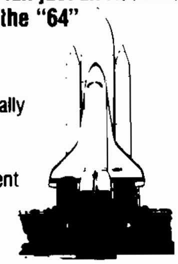
Someday every CBM 64 owner will need a monitor such as this.

Cartridge and Manual — **$24.95**

More than just an Assembler/Editor!
Now for the "64"

MAE

It's a Professionally Designed Software Development System



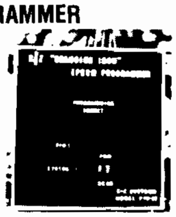
for PET APPLE ATARI
~~$169.95~~
Only $59.95

NOW, The Best for Less!

- Designed to improve Programmer Productivity
- Similar syntax and commands — No need to relearn peculiar syntaxes and commands when you go from PET to APPLE to ATARI.
- Coresident Assembler/Editor — No need to load the Editor then the Assembler then the Editor, etc.
- Also includes Word Processor, Relocating Loader and much more
- Join the ATUG User Group for MAE formatted disks
- STILL NOT CONVINCED? Send for free spec sheet!

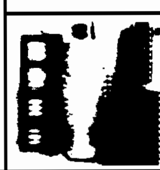
ATARI, PET, AND CBM 64 EPROM PROGRAMMER

Programs 2716 and 2532 EPROMs. Includes hardware and software. PET = $75.00 - ATARI and CBM 64 (both include sophisticated machine language monitor) = $119.95



CBM 64 Debugger

A more sophisticated Machine Language Monitor/Debugger. 20K of object code makes this a powerful tool. Works as a symbolic debugger for the MAE assembler. Diskette and Manual — **$49.95**

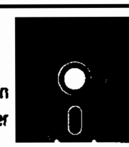


TRAP 65

TRAP 65 is a hardware device that plugs into your 6502's socket. Prevents execution of unimplemented opcodes and provides capability to extend the machines' instruction set.
For PET/APPLE/SYM.
Reduced from $149 95 to $69 95

5¼ INCH SOFT SECTORED DISKETTES

Highest quality. We use them on our PETs, APPLEs, ATARIs, and other computers. $22.50/10 or $44.50/20



Prowriter Printer - Excellent dot matrix print Parallel = $489 00
Serial = $600 00 IEEE = $589 00

DC Hayes Smart Modem = $235 00
DC Hayes Micro Modem II = $289 00

Rana Disk Drive - 375
4 Drive Controller - 114

EPROMS 2716 = $4.50 2532 = $7.50
Over 40 Commodore Programs by Baker (on 4040) = $25.00

Eastern House

3239 Linda Dr.
Winston-Salem, N.C. 27106
(919) 924-2889 (919) 748-8446
Send for free catalog!

VISA · MasterCard

WHILE-DO tests the condition before starting the loop, while FOR-NEXT loops must execute all statements within the loop at least once.

```
WHILE X<Y DO
  PRINT "X IS LESS THAN Y"
  Y := Y-1
ENDWHILE
```

REPEAT-UNTIL, like FOR-NEXT, tests at the bottom of the loop.

```
REPEAT
  INPUT A$
  PRINT "INCORRECT RESPONSE, SHOW A LITTLE RESPECT!"
UNTIL A$="YES SIR"
PRINT "THAT'S BETTER"
```

LOOP ENDLOOP structure can be used to put a test at any place within the loop.

```
INPUT A,B
LOOP
  PRINT A
EXITIF A>10 THEN
PRINT "A REACHED 10 FIRST."
ENDEXIT
  A := A+1
EXITIF Y>10 THEN PRINT "Y REACHED 10 FIRST"
ENDEXIT
  Y := Y+1
ENDLOOP
```

As I get more familiar with BASIC-09, I hope to be bringing more examples of its power, and versatility. It is structured to interface with OS-9, and operates within the path structure that makes OS-9 as powerful as it is.

MICRO

DOSPLUS for Commodore 64

Part 3

Michael J. Keryan

A machine language monitor, a printer formatting program, a repeat key toggle, and a kill (quit DOSPLUS) function for the recently published DOSPLUS utility program.

In the second article in this series (MICRO No. 69), a transient program loader was described. This program will move machine language programs from hidden RAM (located at the same address, or underneath the BASIC ROM) to an area of memory in which they can execute ($C000-$C7FF). In this installment, two such programs are provided: a monitor program and a program that will allow you to format your (non-Commodore) printer to set up character size, top, bottom, left, and right margins, etc. Since we'll need the monitor program to save our other programs, it will be described first.

Add a Machine Language Monitor

A machine language monitor is helpful to move blocks of memory, disassemble ML code, and save ML programs to disk. A very good monitor, probably the most widely used for the Commodore 64, is Jim Butterfield's Supermon64.V1 which can be found on Commodore's Disk Bonus Pack and several public domain disks from the Toronto Pet Users Group. It is a little larger than 2 kilobytes but can be shrunk (to 2K) and relocated to $C000 to be used with DOSPLUS. However, when you think about it, $C000 is the worst place to put the monitor program. You will probably be working on ML programs assembled to run at $C000, so the monitor should be placed elsewhere. As normally configured, Supermon cuts the BASIC user RAM by about 2K, so this is sometimes undesirable.

One place you can put the monitor without taking any user RAM is in the hidden RAM underneath the BASIC ROM. In fact, you can run it from underneath the ROM and even use Kernal ROM subroutines ($E000-$FFFF). However, getting the monitor into this area of memory, establishing the hooks to DOSPLUS and a clean exit to BASIC, and then saving the program to disk is a little tricky, so reread the next sections before jumping in.

Supermon can relocate itself to the top of usable memory; in fact it does this everytime you load and run it. But now you want to relocate it to the top of (hidden) RAM, not user RAM. First, load and run Supermon as usual. You will see on the screen a B* and a register display. Now type in these three lines:

```
.:0001 36
.:0037 00 C0
.G 0880
```

Now you will see another B* and register display, but this time you are running the new version of the monitor located in RAM under the BASIC ROM. What goes here?

The first line above switches out the BASIC ROM by placing $36 into location $0001, the 6510 CPU's memory management register. The second line places $C000 into the top of memory pointers located at $0037, $0038. So instead of your user memory ending at $A000, it now ends at $C000. Next a jump to $0880--this causes the entire monitor program to be relocated. You can find the beginning of the new version to be $B7ED, the end to be $BFFF.

Now, while you are still in the monitor program, assemble the code shown in Listing 1, starting at $C000. This is a boot program that will allow you to jump to the monitor and exit cleanly to BASIC. The program switches out BASIC ROM (as we just did), moves part of itself to $02A7 (an unused portion of memory), sets up the exit vector, and then jumps to the monitor ($B7ED). The exit routine sets the character color to black, switches the BASIC ROM back in, adjusts the stack, then does a jump to the warm start.

This routine now must also be moved to hidden RAM and new table pointers established. Type in four more lines:

```
.T C000 C030 B700
.:A013 01
.:A033 B7
.S " S.ML",08,A000,C000
```

This transfers the small program to $B700 and sets up the tables for a program of 1 block length, starting at $B700, to be moved to $C000 by the transient relocator when a RESTORE, S sequence of keys is used.

To use the monitor with DOSPLUS, the DOSPLUS boot program must be updated to include S as an active key by adding the following line:

```
3065 POKE 52179,89: POKE 52211,207
```

Also add the following line to load the monitor program:

```
1005 IF A1 THEN A2: LOAD "S.ML",8,1
```

Once the monitor is loaded and running, the area of memory from $C000 to $C7FF is free for your use.

Format Your Printer

The first article in this series gave a machine language routine to dump the screen to a printer configured as device #4. Another routine was provided to turn on or off the printer so all output could also be directed to the printer. These programs work with Commodore's 1525 printer or any other printer that can emulate the Commodore printer through an intelligent interface.

Many owners of Commodore 64's have opted to buy printers made by other manufacturers, such as Epson, OKI, Gemini, C. Itoh, or NEC. These printers are not only more expensive than Commodore's 1525, but require an interface costing from $50 to $120 which plugs between the computer and the printer. The main purpose of the interface is to emulate the 1525 printer, i. e. translate certain character codes sent out on the Commodore 64 serial bus to the correct ASCII characters and control codes in the parallel format recognizable by the printer.

Why would so many sane individuals go to such trouble and extra expense to hook up a non-Commodore printer through an interface when the result is a printer that emulates the less expensive Commodore 1525? The answer is that the other printer/interface combinations provide a number of features not available on the 1525. These include faster printing, better quality print, several thousand character buffers, the ability to print on labels, and quite a few special formatting commands that are printer-specific and/or interface-specific. But how many of these special formatting commands are routinely used? Probably none, because to do so requires searching through manuals and using special OPEN and PRINT# commands.

To make these functions easy to use, a printer formatting program is shown in the assembler Listing 2. It can be used as is for an NEC 8023 (or similar printers such as C. Itoh 8510, Prowriter, etc.) and a Tymac Connection printer interface connected to the Commodore 64. This program will require modifications for other combinations of printers/interfaces. The machine language program is designed to be used with DOSPLUS code published in previous articles; it uses a number of DOSPLUS routines.

Entry to the program first saves the current screen, then sets the default set up parameters. You are then instructed to enter one of the following letters:

D for Default set-up

C to Change the set-up (to other than default)

N for No set-up

Any other response is ignored. The default set-up is 12 characters/inch, a left margin of 10 characters, right margin of 6 characters (this gives 80 characters/line with margins), form set-up with skip-over-perf, and non-enhanced print.

If the default set-up is chosen, the following optional input routines are skipped. A choice of C will allow a variety of format set-ups. First is character widths:

Enter	Char/Inch	Char/Line
1	17	132
2	12	96
3	Proportional	96
4	10	80
5	8.5	66
6	6	48
7	Proportional	48
8	5	40

(Proportional Char/Line is approximate)

The next two options are desired number of characters for the left and right margins. These two numbers are subtracted from the number of characters for the full line to get the actual number of characters that can be printed on a line. This is printed out to the screen and you are asked if this is OK. If not, you can then go through the procedure again. If you made a mistake and get a zero or negative line width, an error message is printed and you then have to re-enter the data.

The next option is the fold mode available with the Connection interface. This mode keeps the printer from printing part of a word on one line and the rest of the word on the next. This mode makes BASIC listings more readable. The form length option is next. It sets up the height of the sheet of paper to 66 lines so that a form-feed will get you to the same place on each page. Next is the skip-over-perf option, which allows top and bottom margins to be automatically set up for each page printed. Once set, all pages are formatted in the same manner unless the printer is reset. The last option is the enhanced mode, in which everything is printed with double-striking.

After all the desired parameters have been chosen, the printer port (device #4) must be opened, the desired data output, then the port closed. Actually, this sequence is done three times as described below. First, the port is opened with a secondary address of zero. Codes equivalent to the following are then sent to the interface (they never get to the printer):

CHR$(27)"W"CHR$(0)

This de-activates the Width function of the interface (which normally defaults to 80). This step is required to eliminate an undesired carriage return being sent to the printer later on.

Next the printer port is opened with a secondary address of 6. This is the Connection's transparent mode so all data will be sent directly (unchanged) to the printer. A three byte sequence is output to set up character size. Two more bytes select either normal or enhanced mode. If desired, a 136 byte sequence is output to set up the form length in the printer (with or without skip-over-perf).

Now the printer port is opened with a secondary address of zero again. Then the equivalent of the following is sent to the interface:

CHR$(27)"F"CHR$(x)
CHR$(27)"I"CHR$(y)
CHR$(27)"W"CHR$(z)

where x0 for fold off, x1 for fold on, ynumber of characters for the left margin, and zthe sum of y and the actual line width. Finally a carriage return is sent to activate the margins. The printer port is left either open or closed as desired and then the old screen is restored.

The printer formatting program is assembled to run at $C000. It resides under the BASIC ROM, however, and is transferred to upper memory as described in the last article in this series. Saving the program to disk is a little tricky, so proceed as follows.

With DOSPLUS (including the monitor) in memory, place the printer utility program into the RAM area of

$C000-$C4FF. Use either an assembler/loader or a BASIC loader that POKEs DATA into memory. Then move the program into hidden RAM. This is done easily in BASIC immediate mode:

FOR I 0 TO 1279: A PEEK(I49152): POKE (I41216): NEXT I
POKE 40966,5: POKE 40998,161

The last line sets up the tables for program length (5 blocks) and starting location ($A100).

Now, activate the monitor program by RESTORE, then S. Save the whole 8K block (this now includes the monitor and the printer formatter):

.S " SF.ML",08,A000,C000

Repeat

Normally, only the space bar and the cursor keys of the Commodore 64 have auto-repeat. Holding down any other key gives you only one character for every key press. A flag for the repeat function is located at $028A. The RESTORE, R sequence of keys has been assigned to toggling the repeat mode on and off. When the repeat mode is 'on', all keys will have the same auto-repeat ability. The code for this function is extremely simple, as shown in Listing 3. It fits into an unused area of memory located at $C807. It is only 9 bytes long.

Kill

To activate the DOSPLUS routines you have to load and run the DOSPLUS boot program. To kill it requires either shutting off the computer or a jump to the system reset (SYS 64759). Since it is nearly impossible to remember the number to SYS to, a Kill function was added to DOSPLUS. Hit RESTORE, K to reset the computer, kill DOSPLUS and the wedge, and wipe out any programs in memory.

Wrap-up

In this article, we've added four more functions to DOSPLUS: F to format the printer, S for Supermon, K for kill, and R for repeat on/off. A new BASIC boot program will be printed in the next article in this series. Also provided next time will be a method to store BASIC programs in hidden RAM, so you can switch back and forth between two BASIC programs by hitting a couple of keys.

Listing 1

```
                    ; KERYAN DOSPLUS 3  MARCH 1984
                    ;SUPERMON BOOT -- TO RUN A VERSION
                    ;OF SUPERMON FROM UNDER THE BASIC
                    ;ROMS -- THIS PROGRAM RESIDES AT
                    ; $B700 BUT IS TRANSFERRED TO AND
                    ;RUNS AT $C000 WITH DOSPLUS
                    ;
C000                         ORG $C000
                    ;
C000 A9 36          SPRBOO   LDA #$36        ;TAKE OUT
C002 85 01                   STA $01         ;BASIC ROM
C004 A2 11                   LDX #$11
C006 BD 1C C0       LOOPC    LDA EXTCOD-1,X  ;TRANSFER
C009 9D A6 02                STA $02A6,X     ;CODE BELOW
C00C CA                      DEX             ;TO $02A7
C00D D0 F7                   BNE LOOPC
C00F A9 A6                   LDA #$A6        ;CHANGE EXIT
C011 8D D0 BF                STA $BFD0       ;ROUTINE OF
C014 A9 02                   LDA #$02        ;NEW VERSION
C016 8D D1 BF                STA $BFD1       ;OF MONITOR
C019 4C ED B7                JMP $B7ED       ;GO TO MONITOR
C01C 00                      BRK
C01D A9 90          EXTCOD   LDA #$90        ;THIS CODE IS
C01F 20 D2 FF                JSR $FFD2       ;MOVED TO
C022 A2 37                   LDX #$37        ; $02A7 TO RUN
C024 86 01                   STX $01         ;(SEE BELOW)
C026 AE 3F 02                LDX $023F
C029 9A                      TXS
C02A 6C 02 A0                JMP ($A002)
                    ;
                    ;       ORG $02A7
                    ; EXTCOD LDA #$90  ;OUTPUT BLACK
                    ;       JSR $FFD2 ;COLOR CODE
                    ;       LDX #$37  ;SWITCH IN THE
                    ;       STX $01   ;BASIC ROM
                    ;       LDX $023F ;ADJUST STACK
                    ;       TXS            ;THEN JUMP
                    ;       JMP ($A002) ;TO WARM START365
                    ;
                    ;SET UP THE FOLLOWING BY LOADING
                    ;SUPERMON.  THEN ENTER FOLLOWING:
                    ;
                    ;  .:0001 36     (BASIC ROM OUT)
                    ;  .:0037 00 C0  (FAKE TOP OF MEM)
                    ;  .G 0880       (TO RELOCATE)
                    ;
                    ;NOW SUPERMON IS RELOCATED TO
                    ; $B7ED AND YOU ARE RUNNING IT.
                    ;NOW ENTER THE ABOVE CODE AT $C000
                    ;THEN ENTER THE FOLLOWING:
                    ;
                    ;  .T C000 C030 B700
                    ;  .:A013 01
                    ;  .:A033 B7
                    ;  .S " SM.ML",08,A000,C000
                    ;
                    ; THIS TRANSFERS THE CODE AT $C000
                    ; TO $B700 AND PLACES DATA IN
                    ; TABLE AT A000 (1 BLOCK AND B7
                    ; IS THE STARTING BLOCK FOR S KEY)
                    ;THEN SAVE THE NEW CODE (ALL 8K)
C02D                         END
```

Listing 2

```
C000                     ORG $C000
                ;
00FD            NUML     EQU $FD
00FE            NUMH     EQU $FE
007C            FPFILE   EQU $7C
C88F            D4       EQU $C88F         ;DOSPLUS ROUTINES
C8C3            TABCON   EQU $C8C3
C9BC            PRNTON   EQU $C9BC
C9C4            PRNTOF   EQU $C9C4
CB41            MESSAG   EQU $CB41
CF99            SCRSAV   EQU $CF99
CFB1            SCRRCL   EQU $CFB1
F1CA            OLDOUT   EQU $F1CA         ;KERNAL ROUTINES
FFBA            SETLFS   EQU $FFBA
FFBD            SETNAM   EQU $FFBD
FFC0            OPEN     EQU $FFC0
FFC3            CLOSE    EQU $FFC3
FFC9            CHKOUT   EQU $FFC9
FFCC            CLRCHN   EQU $FFCC
FFCF            CHRIN    EQU $FFCF
FFE4            GETIN    EQU $FFE4
                ;
C000 20 99 CF   FRMPTR   JSR SCRSAV        ;MAKE SURE
C003 A9 45               LDA #$45          ;DEFAULT IS
C005 8D 59 C4            STA TMODES+1      ;ESTABLISHED
C008 A9 0F               LDA #$0F          ;IN CASE OF
C00A 8D 5A C4            STA TMODES+2      ;RE-ENTRY
C00D A9 22               LDA #$22
C00F 8D 5C C4            STA TMODES+4
C012 A9 43               LDA #$43
C014 8D E4 C4            STA TFREND
C017 A9 00               LDA #$00
C019 8D 67 C4            STA TCONNT+2
C01C A9 0A               LDA #$0A
C01E 8D 6A C4            STA TCONNT+5
C021 A9 5A               LDA #$5A
C023 8D 6D C4            STA TCONNT+8
C026 A9 01               LDA #$01
C028 8D 54 C4            STA FORMLN
C02B 20 41 CB   OPENMS   JSR MESSAG
C02E 93 20 20            BYT $93,$20,$20,$12
C032 53 45 54            ASC 'SET-UP FOR NEC/
C041 50 52 4F            ASC 'PROWRITER--CONNECTION'
C056 92 0D 0D            BYT $92,$0D,$0D
C059 20 44 45            ASC ' DEFAULT: CPI=12 LM=10'
C06F 20 52 4D            ASC ' RM=6 LINE=80.'
C07D 0D                  BYT $0D
C07E 20 46 4F            ASC ' FORM=YES SKIP=YES'
C090 20 45 4E            ASC ' ENHANCE=NO.'
C09C 0D 0D               BYT $0D,$0D
C09E 45 4E 54            ASC 'ENTER:'
C0A4 0D                  BYT $0D
C0A5 20 20 20            ASC '     <D> DEFAULT SET-UP'
C0BB 0D                  BYT $0D
C0BC 20 20 20            ASC '     <C> CHANGE SET-UP'
C0D1 0D                  BYT $0D
C0D2 20 20 20            ASC '     <N> NO SET-UP'
C0E3 0D                  BYT $0D
C0E4 20 20 20            ASC '     ?'
C0E9 00                  BYT $00
C0EA 20 E4 FF   WATDCN   JSR GETIN
C0ED C9 00               CMP #$00
C0EF F0 F9               BEQ WATDCN
C0F1 C9 4E               CMP #$4E          ;N
C0F3 D0 03               BNE DKEY
C0F5 4C 9A C3            JMP NOSET
C0F8 C9 44      DKEY     CMP #$44          ;D
C0FA D0 03               BNE CKEY
C0FC 4C 46 C3            JMP OUTALL
C0FF C9 43      CKEY     CMP #$43          ;C
C101 F0 03               BEQ LINWID
C103 4C 2B C0            JMP OPENMS
C106 20 41 CB   LINWID   JSR MESSAG
C109 0D 0D               BYT $0D,$0D
C10B 43 48 41            ASC 'CHARACTERS'
C115 0D                  BYT $0D
C116 2F 4C 49            ASC '/LINE /INCH  ENTER'
C128 0D                  BYT $0D
C129 20 31 33            ASC ' 132    17    <1>'
C13A 0D                  BYT $0D
C13B 20 20 39            ASC '  96    12    <2>'
C14C 0D                  BYT $0D
C14D 20 20 39            ASC '  96   PROP.  <3>'
C15E 0D                  BYT $0D
C15F 20 20 38            ASC '  80    10    <4>'
C170 0D                  BYT $0D
C171 20 20 36            ASC '  66    8.5   <5>'
C182 0D                  BYT $0D
C183 20 20 34            ASC '  48     6    <6>'
C194 0D                  BYT $0D
C195 20 20 34            ASC '  48   PROP.  <7>'
C1A6 0D                  BYT $0D
C1A7 20 20 34            ASC '  40     5    <8>'
C1B8 0D                  BYT $0D
C1B9 20 20 20            ASC '
C1C6 00                  BYT $00
C1C7 20 41 CB   QUESTM   JSR MESSAG
C1CA 20 3F               ASC ' ?'
C1CC 00                  BYT $00
C1CD 20 21 C4   LINMOD   JSR GETNBR
C1D0 C9 09               CMP #$09          ;>8?
C1D2 B0 F3               BCS QUESTM        ;YES, RETRY
C1D4 C9 01               CMP #$01          ;>0?
C1D6 90 EF               BCC QUESTM        ;NO, RETRY
C1D8 AA                  TAX
C1D9 BD 5C C4            LDA TCHRLN-1,X
C1DC 8D 52 C4            STA CHRIND
C1DF E0 01               CPX #$01
C1E1 F0 04               BEQ CONDEN
C1E3 E0 05               CPX #$05
C1E5 D0 05               BNE PICA
C1E7 A9 51      CONDEN   LDA #$51
C1E9 8D 59 C4            STA TMODES+1      ;17 CPI MODE
C1EC E0 04      PICA     CPX #$04
C1EE F0 04               BEQ TENCHR
C1F0 E0 08               CPX #$08
C1F2 D0 05               BNE PROP
C1F4 A9 4E      TENCHR   LDA #$4E
C1F6 8D 59 C4            STA TMODES+1      ;10 CPI MODE
C1F9 E0 03      PROP     CPX #$03
C1FB F0 04               BEQ PROPOR
C1FD E0 07               CPX #$07
C1FF D0 05               BNE ELITE
C201 A9 50      PROPOR   LDA #$50
C203 8D 59 C4            STA TMODES+1      ;PROPORTIONAL
C206 E0 05      ELITE    CPX #$05          ;MODE <5?
C208 90 05               BCC LEFT          ;YES GO ON
```

```
C20A A9 0E              LDA #$0E        ;NO WIDE MODE
C20C 8D 5A C4           STA TMODES+2
C20F 20 41 CB   LEFT    JSR MESSAG
C212 0D 0D              BYT $0D,$0D
C214 23 20 43           ASC '# CHAR LEFT MARGIN ?'
C228 00                 BYT $00
C229 20 21 C4           JSR GETNBR
C22C 8D 6A C4           STA TCONNT+5    ;LEFT STOP
C22F 20 41 CB           JSR MESSAG
C232 0D 0D              BYT $0D,$0D
C234 23 20 43           ASC '# CHAR RIGHT MARGIN ?'
C249 00                 BYT $00
C24A 20 21 C4           JSR GETNBR
C24D 8D 53 C4           STA RITMAR
C250 AD 52 C4           LDA CHRIND
C253 38                 SEC
C254 ED 53 C4           SBC RITMAR
C257 B0 1C              BCS RWIDTH      ;LINE >0 CHAR
C259 20 41 CB   NEGLIN  JSR MESSAG
C25C 0D 0D              BYT $0D,$0D
C25E 4E 4F 20           ASC 'NO GOOD. TRY AGAIN.'
C271 00                 BYT $00
C272 4C 06 C1           JMP LINWID
C275 8D 6D C4   RWIDTH  STA TCONNT+8    ;RIGHT STOP
C278 ED 6A C4           SBC TCONNT+5
C27B 90 DC              BCC NEGLIN
C27D 85 FD              STA NUML        ;LINE WIDTH
C27F 20 41 CB           JSR MESSAG
C282 0D 0D              BYT $0D,$0D
C284 54 48 49           ASC 'THIS GIVES A LINE OF '
C299 00                 BYT $00
C29A A9 00              LDA #$00
C29C 85 FE              STA NUMH
C29E A9 60              LDA #$60        ;CHANGE CODE
C2A0 8D C0 CB           STA TABCON-3    ;TO SUBROUTINE
C2A3 20 8F CB           JSR D4
C2A6 A9 4C              LDA #$4C        ;RESTORE D4
C2A8 8D C0 CB           STA TABCON-3    ;CODE
C2AB 20 41 CB           JSR MESSAG
C2AE 20 43 48           ASC ' CHARACTERS'
C2B9 0D                 BYT $0D
C2BA 4F 4B              ASC 'OK'
C2BC 00                 BYT $00
C2BD 20 E3 C3           JSR GETYN       ;SO FAR OK?
C2C0 90 03              BCC FOLDMD      ;YES GO ON
C2C2 4C 06 C1           JMP LINWID
C2C5 20 41 CB   FOLDMD  JSR MESSAG
C2C8 0D 0D              BYT $0D,$0D
C2CA 57 41 4E           ASC 'WANT FOLD MODE ON'
C2DB 00                 BYT $00
C2DC 20 E3 C3           JSR GETYN
C2DF B0 05              BCS FORMSU
C2E1 A9 01              LDA #$01        ;FOLD MODE
C2E3 8D 67 C4           STA TCONNT+2    ;SET SWITCH
C2E6 20 41 CB   FORMSU  JSR MESSAG
C2E9 0D 0D              BYT $0D,$0D
C2EB 53 45 54           ASC 'SET UP FORM LENGTH'
C2FD 00                 BYT $00
C2FE 20 E3 C3           JSR GETYN
C301 90 07              BCC SKIPPF
C303 A9 00              LDA #$00        ;NO FORM SO
C305 8D 54 C4           STA FORMLN      ;RESET SWITCH
C308 F0 1E              BEQ ENHAN       ;ALSO SKIP SKIP
C30A 20 41 CB   SKIPPF  JSR MESSAG
C30D 0D 0D              BYT $0D,$0D
C30F 53 4B 49           ASC 'SKIP OVER PERF'
C31D 00                 BYT $00
C31E 20 E3 C3           JSR GETYN
C321 90 05              BCC ENHAN
C323 A9 40              LDA #$40        ;DELETE END OF
C325 8D E4 C4           STA TFREND      ;FORM CODE
C328 20 41 CB   ENHAN   JSR MESSAG
C32B 0D 0D              BYT $0D,$0D
C32D 45 4E 48           ASC 'ENHANCED PRINT'
C33B 00                 BYT $00
C33C 20 E3 C3           JSR GETYN
C33F B0 05              BCS OUTALL
C341 A9 21              LDA #$21        ;SET ENHANCED
C343 8D 5C C4           STA TMODES+4    ;MODE
C346 20 C4 C9   OUTALL  JSR PRNTOF      ;PRINTER OFF
C349 A0 00              LDY #$00
C34B 20 C3 C3           JSR FPOPEN      ;SEC ADR=0
C34E A2 00              LDX #$00        ;3 BYTES
C350 BD 55 C4   LOOPF1  LDA TFLYES,X
C353 20 CA F1           JSR OLDOUT      ;THIS MAKES
C356 E8                 INX             ;SURE NO CR
C357 E0 03              CPX #$03        ;IS GENERATED
C359 D0 F5              BNE LOOPF1
C35B 20 DA C3           JSR FPCLOS
C35E A0 06              LDY #$06
C360 20 C3 C3           JSR FPOPEN      ;SEC ADR=6
C363 A2 00              LDX #$00        ;5 BYTES
C365 BD 58 C4   LOOPF2  LDA TMODES,X
C368 20 CA F1           JSR OLDOUT
C36B E8                 INX
C36C E0 05              CPX #$05
C36E D0 F5              BNE LOOPF2
C370 AD 54 C4           LDA FORMLN      ;FORM SET UP?
C373 F0 0D              BEQ FCL         ;NO, SKIP IT
C375 A2 00              LDX #$00        ;136 BYTES
C377 BD 6F C4   LOOPF3  LDA TFORLN,X
C37A 20 CA F1           JSR OLDOUT
C37D E8                 INX
C37E E0 88              CPX #$88
C380 D0 F5              BNE LOOPF3
C382 20 DA C3   FCL     JSR FPCLOS
C385 A0 00              LDY #$00
C387 20 C3 C3           JSR FPOPEN      ;SEC ADR=0
C38A A2 00              LDX #$00        ;10 BYTES
C38C BD 65 C4   LOOPF4  LDA TCONNT,X
C38F 20 CA F1           JSR OLDOUT      ;THIS GOES
C392 E8                 INX             ;TO INTERFACE
C393 E0 0A              CPX #$0A
C395 D0 F5              BNE LOOPF4      ;NOT PRINTER
C397 20 DA C3           JSR FPCLOS
C39A 20 41 CB   NOSET   JSR MESSAG
C39D 0D 0D              BYT $0D,$0D
C39F 57 41 4E           ASC 'WANT THE PRINTER LEFT O'
C3B6 00                 BYT $00
C3B7 20 E3 C3           JSR GETYN
C3BA B0 03              BCS FPTRDN
C3BC 20 BC C9           JSR PRNTON      ;PRINTER ON
C3BF 20 B1 CF   FPTRDN  JSR SCRRCL
C3C2 60                 RTS
                ;
                ;SUBROUTINES TO SUPPORT ABOVE CODE
                ;
C3C3 A9 7C      FPOPEN  LDA #FPFILE
```

```
C3C5 A2 04              LDX #$04
C3C7 20 BA FF           JSR SETLFS
C3CA A9 00              LDA #$00
C3CC 20 BD FF           JSR SETNAM
C3CF 20 C0 FF           JSR OPEN
C3D2 B0 06              BCS FPCLOS      ;ERROR
C3D4 A2 7C              LDX #FPFILE
C3D6 20 C9 FF           JSR CHKOUT
C3D9 60                 RTS
               ;
C3DA A9 7C      FPCLOS  LDA #FPFILE
C3DC 20 C3 FF           JSR CLOSE
C3DF 20 CC FF           JSR CLRCHN
C3E2 60                 RTS
               ;
C3E3 20 41 CB   GETYN   JSR MESSAG
C3E6 20 3C 59           ASC ' <Y/N> ?'
C3EE 00                 BYT $00
C3EF 20 E4 FF   WATNUL  JSR GETIN       ;KEEP FROM
C3F2 C9 00              CMP #$00        ;FALSE ENTRY
C3F4 D0 F9              BNE WATNUL
C3F6 20 E4 FF   WATYN   JSR GETIN
C3F9 C9 00              CMP #$00
C3FB F0 F9              BEQ WATYN
C3FD C9 59              CMP #$59        ;Y FOR YES
C3FF D0 02              BNE NEGCHK
C401 18                 CLC
C402 60                 RTS
C403 C9 4E      NEGCHK  CMP #$4E        ;N FOR NO
C405 D0 DC              BNE GETYN
C407 38                 SEC
C408 60                 RTS
               ;
C409 20 41 CB   NUMERR  JSR MESSAG
C40C 0D 0D              BYT $0D,$0D
C40E 45 52 52           ASC 'ERROR--TRY AGAIN.'
C41F 0D 00              BYT $0D,$00
C421 A9 00      GETNBR  LDA #$00
C423 85 FD              STA NUML
C425 85 FE              STA NUMH
C427 20 CF FF           JSR CHRIN
C42A C9 30      FPDEC   CMP #$30
C42C 90 DB              BCC NUMERR
C42E C9 3A              CMP #$3A
C430 B0 D7              BCS NUMERR
C432 29 0F              AND #$0F
C434 A2 11              LDX #$11
C436 D0 05              BNE ND3
C438 90 02      ND1     BCC ND2
C43A 69 09              ADC #$09
C43C 4A         ND2     LSR
C43D 66 FE      ND3     ROR NUMH
C43F 66 FD              ROR NUML
C441 CA                 DEX
C442 D0 F4              BNE ND1
C444 20 CF FF           JSR CHRIN
C447 C9 0D              CMP #$0D
C449 D0 DF              BNE FPDEC
C44B A5 FE              LDA NUMH
C44D D0 BA              BNE NUMERR
C44F A5 FD              LDA NUML
C451 60                 RTS
               ;
C452 60         CHRIND  BYT $60
C453 06         RITMAR  BYT $06
C454 01         FORMLN  BYT $01
               ;
C455 1B 57 00   TFLYES  BYT $1B,$57,$00
               ;
C458 1B 45 0F   TMODES  BYT $1B,$45,$0F,$1B,$22
               ;
C45D 84 60 60   TCHRLN  BYT $84,$60,$60,$50
C461 42 30 30           BYT $42,$30,$30,$28
               ;
C465 1B 46 00   TCONNT  BYT $1B,$46,$00 ;FOLD OFF
C468 1B 49 0A           BYT $1B,$49,$0A ;LEFT MARGIN
C46B 1B 57 5A           BYT $1B,$57,$5A ;RIGHT STOP
C46E 0D                 BYT $0D         ;CAR RET
               ;
C46F 1D 41      TFORLN  BYT $1D,$41
C471 40 40 40           ASC '@@@@@@@@@@'
C47B 40 40 40           ASC '@@@@@@@@@@'
C485 40 40 40           ASC '@@@@@@@@@@'
C48F 40 40 40           ASC '@@@@@@@@@@'
C499 40 40 40           ASC '@@@@@@@@@@'
C4A3 40 40 40           ASC '@@@@@@@@@@'
C4AD 40 40 40           ASC '@@@@@@@@@@'
C4B7 40 40 40           ASC '@@@@@@@@@@'
C4C1 40 40 40           ASC '@@@@@@@@@@'
C4CB 40 40 40           ASC '@@@@@@@@@@'
C4D5 40 40 40           ASC '@@@@@@@@@@'
C4DF 40 40 40           ASC '@@@@@'
C4E4 43         TFREND  BYT $43         ;BOTTOM MARGIN
C4E5 40 40 40           ASC '@@@@@@@@@@'
C4EF 40 40 40           ASC '@@@@@'
C4F4 41 40 1E           BYT $41,$40,$1E
               ;
C4F7                    END
```

Listing 3

```
               ; KERYAN DOSPLUS+ PART 3 MARCH 1984
               ;REPEAT FUNCTION TOGGLE ON/OFF
               ;
               ;   USE WITH DOS+
               ;
C807                    ORG $C807
               ;
028A            RPTFLG  EQU $028A
               ;
C807 AD 8A 02   REPETG  LDA RPTFLG
C80A 49 80              EOR #$80
C80C 8D 8A 02           STA RPTFLG
C80F 60                 RTS
               ;
               ;CHANGE POINTERS TO USE THE R KEY
               ; FOR REPETG
               ;
CBD2                    ORG $CBD2
CBD2 07                 BYT $07
               ;
CBF2                    ORG $CBF2
CBF2 C8                 BYT $C8
               ;
CBF3                    END
```

Michael J. Keryan may be reached at 713 Locust Drive, Tallmadge, OH 44278.

MICRO™

From Here to Atari

by Paul S. Swanson

Free Connections

In a surprising number of homes all over the country people are donating their time, computer equipment and phone lines to the public. These people operate computer bulletin board services (BBS's). Calling BBS's is becoming a very popular pasttime. These BBS's, or "boards" as they are more commonly called, usually contain a message base where callers can post messages on just about any topic they care to discuss. For example, in the message bases of the boards I call in this area contain ongoing discussions about national politics, US-USSR relations, nuclear weapons control and economics. There are also other discussions concerning the possibilities of colonizing other planets. In addition, there are more "down to earth" topics such as for sale and wanted ads, announcements of the numbers of new boards, questions and answers about computers and critiques of television shows and movies. Other features offered by these boards include on-line games, local weather reports, public domain software, private electronic mail, lists of the telephone numbers of other boards, help screens and chat. The chat mode calls for the BBS's owner, usually referred to as the "SysOp," to converse via the keyboards. Almost all of the boards have most of these features. Some boards also have multiple message bases so that conversations are categorized by some general criteria. If, for example, you didn't want to have anything to do with the current political discussion, you wouldn't have to see it at all, but could look at all of the other messages.

Obtaining the public domain software available on the boards requires that you have a terminal program that can download files. Normally, this requires that you have a disk drive on your system, but there are some programs that will work with cassettes. There are an enormous number of public domain programs on the boards ranging from games to serious utility programs, which makes obtaining a terminal program that allows downloading a worthwhile effort. If you run into problems finding one for your particular computer, you can leave a message on one of the boards asking for help.

This service is, of course, not free of problems. Occasionally there will be a "problem caller" on a board, doing some things which cause problems on the system. For this reason, many boards have instituted password systems. These passwords are almost always free as long as you leave your real name and a valid telephone number for verification. That way, the SysOp knows who is using his computer and if anyone causes problems, he knows who to call by the password used. Although having to apply for a password intimidates some new users, it has been effective in eliminating problems like this on most boards. Usually on these password systems, you are also allowed use of a fictitious name on the board so that, if you are shy about using your own name, only you and the SysOp know who you really are. Getting started is usually the hardest part of communicating with free boards. The problem is getting the first number to call. Not too many places list these free telephone numbers. I have listed some boards below which are in the Cambridge area. North Shore AMIS, which is running on an Atari, has a list of Atari-run boards all over the country. The numbers are in the Features section in a file called ATARIBBS. When you have called there for the list, call the board that is listed as being nearest you to look for the numbers of other boards that are in your area.

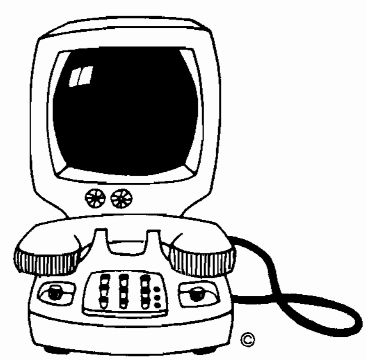
There are some standards to be considered in the communication. Almost all boards, at either 300 or 1200 baud, communicate with 8 bits per word with one stop bit, no parity, full duplex ASCII. If your terminal program has ways to alter the communications parameters, those will probably be the items listed. If any of them are not listed or not alterable, your terminal program probably already assumes the correct value, since those selections are so standard. If you are not sure of any particular setting, try it one way and call a board. If you find that the computers cannot communicate, alter the parameter in question and try again. Trial and error will eventually get you through, although odds are good that the default values set for your program are those described above.

Calling the boards in your area can be informative, useful and, of course, very entertaining. The biggest problem with them is that they are addictive, so you have to keep careful track of how much time you spend on this interesting new hobby. Some BBS's in the Cambridge area (all in the 617 area code) are:

- ☐ *Nite Lite 576-2426 (6pm-6am only)*
- ☐ *The Outpost 259-0181*
- ☐ *North Shore AMIS 595-0211*
- ☐ *Boston Bullet 266-7789*
- ☐ *Boston Bullet TBBS 267-7751*
- ☐ *The Trash Bin 497-6641*
- ☐ *King's Castle 444-5401*

There are actually over 30 free boards which are in the local dialing area of Cambridge. The other numbers are listed on the above boards. All of the boards above will answer at either 300 or 1200 baud except King's Castle, which is 300 baud only. The hours for my board, Nite Lite, are eastern time. Please make the appropriate adjustment for your time zone.

MICRO™



Bezier Curves

A FORTH Implementation

by Richard H. Turpin

The Bezier method allows a curve to be represented with a minimum amount of data. Only four "control points" are needed for this example

When displaying curves and surfaces in computer graphics systems one of the concerns is the method of curve description. A straight line is compactly defined in terms of its end points, and a circle can be defined simply in terms of its center and radius. But how do you define a curve without using a large number of data points?

There are a number of methods for defining curves efficiently, one of which is the Bezier curve. In this article FORTH code is described for drawing Bezier curves, given four "control points." An application program, also in FORTH, is included to illustrate the use of Bezier curves, along with other graphics primitives, to generate line drawings.

Bezier Curves Defined

The purpose of the Bezier method is to represent a curve with a minimum amount of data. It is an interpolation scheme which uses "control points" to define a curve in two dimensional space. Each control point is an X,Y data pair defining a point in the X,Y plane.

The Bezier Curve is defined by the equation[1]

$$(1) \quad P(t) = \sum_{i}^{n} P_i J_{n,i}(t)$$

$$0 <= t <= 1$$

P_i, $i=0,1,...,n,$

$J_{n,i}(t)$ $\qquad$ $J_{n,i}(t)$

where the P(i), i=0,1,...,n, are the control points. The functions J(n,i) (t) are called "blending functions" because they act to blend the effects of all the control points in determining each point on the curve. J(n,i)(t) is defined by equations 2 and 3.

$$(2) \quad J_{n,i}(t) = \binom{n}{i} t^i (1-t)^{n-i}$$

where

$$(3) \quad \binom{n}{i} = \frac{n!}{i!(n-i)!}$$

The parameter **t** provides a single variable for representing the curves in place of the two natural variables **X** and **Y**. This "parameterization" of the curve results in a simplification of the equations used in representing the curve, and hence in curve generation. The quality of the resultant curve is enhanced, also.

A common application of equations 1, 2 and 3 is derived by letting n = 3. The result is a cubic Bezier curve which is defined by four control points. The four control points P0, P1, P2 and P3 define a curve in the following way. P0 and P3 define the end points of the curve. P1 and P2 control the shape of the curve,but they do not lie on the curve. Points P0 and P1 determine the direction in which the curve leaves P0, while P2 and P3 determine the angle of arrival at P3. Figure 1 provides example Bezier curves to demonstrate this.

Raise your Apple's IQ Twelve Times A Year!

Try a NIBBLE!

Here's what some of our Readers say:

- ☐ *"Certainly the best magazine on the Apple!"*
- ☐ *"Impressed with the quality and content."*
- ☐ *"Programs remarkably easy to enter."*
- ☐ *"I'll be a subscriber for life!"*
- ☐ *"Your service is fantastic . . . as a matter of fact, I'm amazed!"*

Try a NIBBLE!

NIBBLE is focused completely on the Apple and Applesoft-compatible computers.

Buy NIBBLE through your local Apple Dealer or subscribe now with the Coupon or Order Card in this issue.

You'll want Back Issues Too!

Here are some examples of programs you can get:

The Investor—Stock Tracking, Reporting, and Graphing.

Recipe Box—Kitchen/Menu Management made Fun.

The Librarian—Auto Logging and Retrieval of your Disks.

Designer/Illustrator—Art/Design Creation and Composition with Graphics.

Machine Language Editor—Quick and Easy Aid for Typing and Changing M/L Programs.

And Much . . . Much More!

NIBBLE will become a permanent part of your Reference Library. Discover why 95% of NIBBLE Readers save every issue!

Join more than 120,000 Apple/Ace users who say: "NIBBLE IS TERRIFIC!"

SUBSCRIBE NOW AND SAVE $12.00 OFF THE COVER PRICE!

A One Year Subscription Brings You 12 Issues With:

Over $500 of Programs for your Home, Business, Education and Entertainment. Complete Program Listings with Instructions.

Comprehensive Articles that show what each program does, how to use it and how to type it into your Apple, Franklin ACE or other Applesoft-compatible computer.

Regular Features for the Beginner and Expert.

On The Scene
The Latest New Software/Hardware Releases.

Products! Inside and Out
Comprehensive Product Reviews.

Education Corner
Programs that help make Learning Fun.

Tips 'N Techniques
Little known programming Tricks you can Use.

Disassembly Lines
An Expert reveals the mysteries of Applesoft.

Utilities
Superchargers for Basic, DOS, Printing, and More.

Games
Arcade Fun you can Type and Run.

Note
- ☐ Domestic U.S. First Class subscription rate is $51.95
- ☐ Canada Air Mail subscripton rate is $59.95
- ☐ Outside the U.S. and Canada Air Mail subscription rate is $89.95

All payments must be in U.S. funds drawn on a U.S. bank.

nibble

MasterCard VISA
We accept Master Charge & Visa

Box 325, Lincoln, MA 01773 (617) 259-9710

I'll try nibble!
Enclosed is my $26.95 (for 12 issues)
(Outside U.S., see special note on this page.)

☐ check ☐ money order ☐ bill me (U.S. only)

Your subscription will begin with the next issue published after receipt of your check/money order.

Card # ____________________ Expires __________
PLEASE PRINT CLEARLY
Signature ____________________
Name ____________________
Address ____________________
City ____________________
State ____________________ Zip __________

Notice in each example that the curves leave P0 and approach P3 in directions defined by the straight lines connecting P1 to P0 and P2 to P3, respectively. In addition to the influence P1 and P2 have on the curve at the endpoints, they have considerable control on the overall shape of the curve. The examples given in Figure 1 illustrate the wide range of curve shapes which can be generated.

In Figure 2 is an example of a family of curves which is easily generated by changing one or more of the control points.

A number of Bezier curves can be used to form complex curves and surfaces, such as shown in Figure 3.

Implementation of Bezier Curves in FORTH

From equations 1, 2 and 3, with n = 3, the equation for a cubic Bezier curve can be written as in equation 4,

$$(4)\quad P(t) = (1-t)^3*P0 + 3*t*(1-t)^2*P1 + 3*t^2*(1-t)*P2 + t^3*P3$$

$$0<=t<=1$$

where each point P(t) represents an X,Y position on a coordinate system. Note that for t = 0, P(0) = P0 and for t = 1, P(1) = P3. For values of t between 0 and 1 P(t) is a blending of the four points P0, P1, P2 and P3.

Equation 4 could be programmed as it is, but a more efficient representation can be found. It can be rewritten as

$$(5)\, P(t) + t^3*[(P3-P0)-3*(P2-P1)] + t^2*[3*(P2-P1)-3*(P1-P0)] + t*[3*(P1-P0)] + P0$$

Letting

$$(6)\quad P0' = P0$$

$$(7)\quad P1' = 3*(P1-P0)$$

$$(8)\quad P2' = 3*(P2-P1)-3*(P1-P0)$$

$$(9)\quad P3' = (P3-P0)-3*(P2-P1)$$

equation 5 becomes

$$(10)\quad P(t) = t^3*P3' + t^2*P2' + t*P1' + P0'$$

or

$$(11)\quad P(t) = t*(t*(t*P3'+P2')+P1')+P0'$$

Note in equation 11 the repeated expression (t*Pm + Pn) found nested three levels deep. This expression for P(t) requires considerably fewer calculations than the original form in equation 4. In particular, the number of multiples is reduced to six (three for each dimension X and Y). The computations defined in equations 6 through 9 are performed once for a given curve. To generate the curve, equation 11 is then evaluated for several values of **t**.

Figure 1. Example Bezier Curves

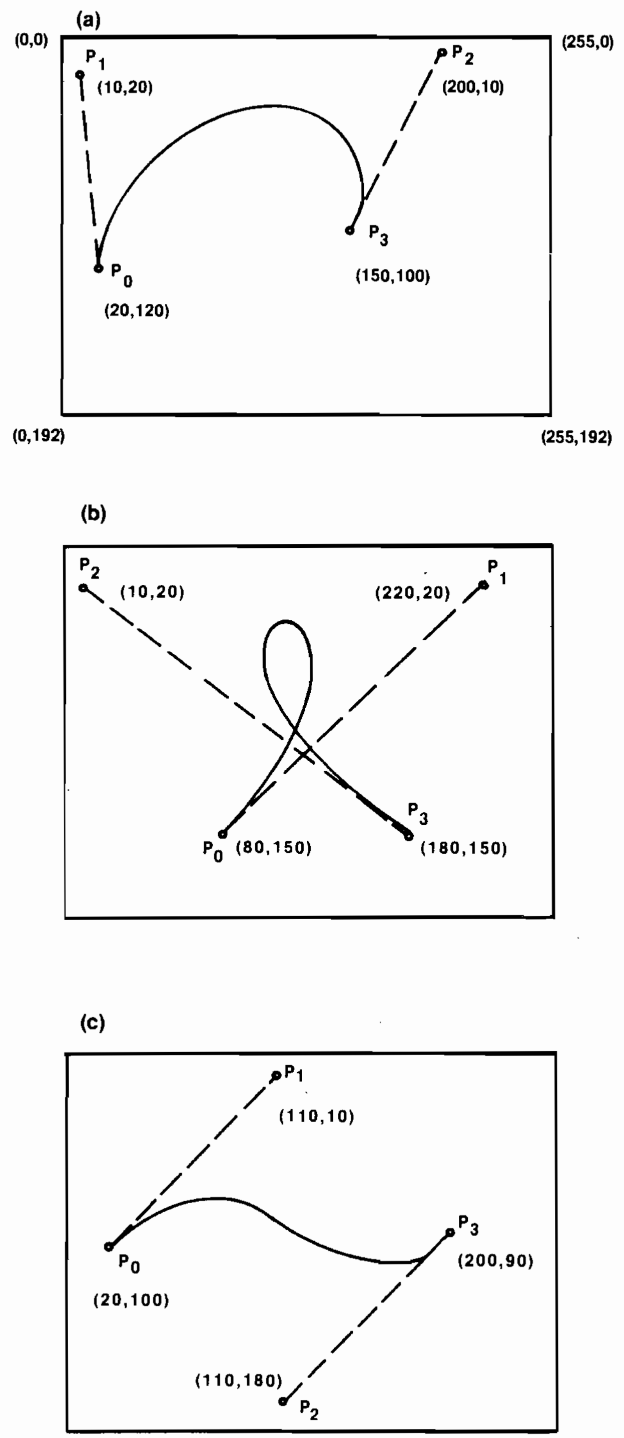


Figure 2. A Family of Bezier Curves

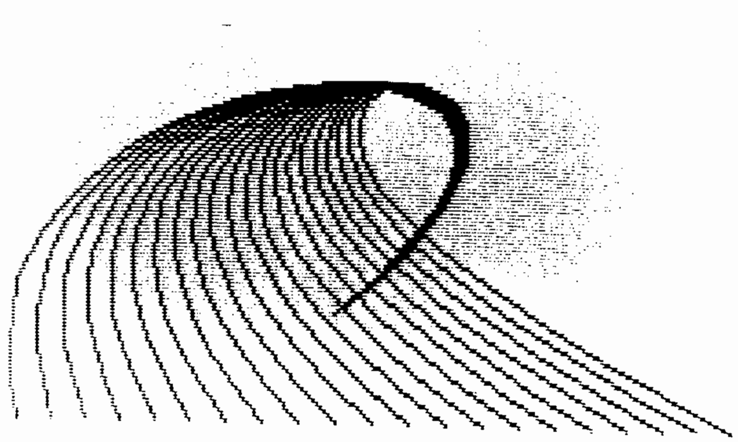
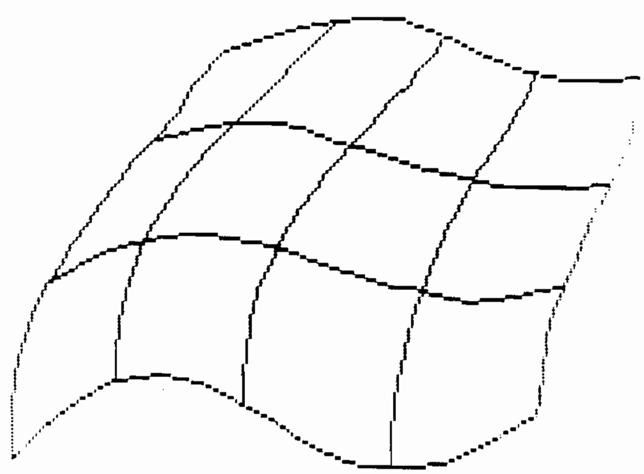


Figure 3. Generating Complex Curves and Surfaces



FORTH screens 42 through 47 given in Listing 1 are an implementation of equation 11. Taking advantage of FORTH extensibility, X,Y point operators P@, P!,P+, P-, P*, P*/, PDUP and PSWAP are defined (see screens 43 and 44) corresponding to the FORTH operators @ (fetch), ! (store, = (plus), - (minus),* (times), */ (times-divide), DUP (duplicate top of stack), and SWAP (exchange top two items on stack). {Note: the notation (m,n --- p,q) defines stack contents before (on the left) and after (on the right) the word executes.} Control point storage is defined in terms of point names P0, P1, P2 and P3, rather than (X0,Y0), (X1,Y1), etc., for convenience. In each case, the point name Pi points to the corresponding X value; the Y value is stored in the next cell. (See lines 7-11 of screen 44.)

To draw the Bezier curve, points P(t) are calculated for several values of t and the points are connected by straight lines. By choosing small enough increments in **t**, the resultant plot appears as a "smooth" curve (as a function of the resolution of the graphics system). In the implementation presented here integer arithmetic is used, not only because FORTH arithmetic is integer, but also because it is faster. The parameter **t** must be scaled so that it takes on only integer values. A variable **N** is defined as the upper limit of **t. N** also specifies the number of line segments used to build the curve. To generate a curve, then, the graphics cursor is moved to the beginning of the curve, P(0) = P0, then P(T) is computed for T = 1 and a line is drawn to P(1). P(T) is computed for T = 2 and a line is drawn from P(1) to P(2), etc. The process is repeated to generate **N** line segments and to end at point P(N) = P3.

The number of segments is determined by computing P(t) at the curve midpoint, i.e., at T = N/2. The maximum of

$$X=(\,|X(N/2)-X(0)|\,)/3$$

and

$$Y=(\,|Y(N/2)-Y(0)|\,)/3$$

is taken as the segment count, with a minimum of 3 and a maximum of 50. These minimum and maximum values are somewhat arbitrary. They affect both curve quality and curve generation speed. The FORTH word FIX.N defined in screen 46 performs this calculation and sets the value of **N**. The variable N.SC, defined in screen 44 provides a convenient means to experiment with the segment count. Its default value is 3 (see line 14, screen 44).

Equations 6 through 9 are implemented in the word COMPUTE.P' in screen 45. The word STORE.POINTS is used to define the control points P0, P1, P2 and P3 by writing data from the stack into the corresponding storage locations. LIST.POINTS and SHOW.POINTS are utilities for viewing the control points on the CRT or on the graphics display device.

Only two graphics commands are used to draw the curve. They are TMOVE, which moves the cursor (without drawing) to the specified X,Y position on the screen, and TDRAW, which draws a line from the present cursor position to the specified position. Equivalent commands should be readily defined for most graphics systems if they do not already exist.

MOVE.TO.P0, defined in screen 46, simply moves the graphics cursor to control point P0, the beginning of the curve. LOAD.POINTS was defined to load the modified control points P0'-P3' onto the stack for use by the word P(T). P(T) is an implementation of equation 11, the simplified expression derived earlier. The word performs the general computation (t*Pm + Pn) three times as defined by equation 11.

The parameter, **T**, is varied over its range of 1 to N in the word CURVE (screen 47), which generates the Bezier curve. BEZIER puts everything together to produce a curve given a set of four control points P0, P1, P2 and P3.

To generate the example curve shown in Figure 1b, the following was executed.

```
80 150 220 20 10 20
180 150 STORE.POINTS
BEZIER
```

For the family of curves given in Figure 2 the word given below was defined, then executed:

```
: FAMILY 240 30 DO I
  P0 ! BEZIER 10 +LOOP ;

20 150 1 40 240 20 120
  120 STORE.POINTS
  FAMILY
```

where "I P0 !" modifies only X0 (not Y0) since X0 is stored at the address defined by P0. To modify Y0 we would use P0 2 + !. We could also use the point store word P! defined in screen 43 to redefine any one of the four points. For example, "50 140 P0 P!" would change point P0 X and Y values to 50 and 140 respectively.

An Application: Line Drawings

Screens 35-39 (listing 2) define an application of the Bezier curve code, along with other graphics primitives, to generate line drawings from a table of data. Nine commands are defined, as listed in screen 35. They provide the ability to build a picture from dots, straight lines, rectangles, circles and, of course, Bezier curves. The data are stored on disk in FORTH screens so that it can be generated, edited and listed using the FORTH editor.

The only nonstandard words used are CASE (screen 38), and the graphics primitives COLOR, TDOT, TCIRCLE, TMOVE, TDRAW, TRECT and CLEAR (used in screens 36 and 37). CASE is the version written by C.E. Eaker and A.J. Monroe, published in FORTH Dimensions.[2] The graphics primitives are a subset of the primitives supported by the author's system (a TMS9918A-based color graphics system). A similar set of primitives should be available or could be written for other graphics systems.

The application and examples were written assuming a FORTH disk format of 1K bytes per sector so that one sector stores an entire screen of data ("screen" here refers to the FORTH screen, not the graphics screen). The word BLOCK (screen 36, line 5) will then load an entire screen. Other configurations may require slight modification of the data file and/or the code.

READ.BLK in screen 36 loads in the specified data file and establishes it as the source of data for READ.COMMAND and DATA. It also displays at the terminal the figure title or subtitle. READ.COMMAND (screen 39) reads the next character from the data file and expects it to be the ASCII code of one of the nine commands. CASE, used in EXEC.COMMAND (screen 38), then selects the appropriate word to execute that command.

The word DATA in screen 36 is a key word in the remaining word definitions. It retrieves a parameter from the data file. For example, in the definition of the word DRAW.CURVE it is used eight times to load four X,Y data pairs from the file.

Each screen of a data file begins with a title or subtitle which is terminated by a ")". Commands are given one after the other, each with the required number of data values. At least one space must separate all commands and data values. The "E" command terminates the data file. The "N" command permits building large data files by linking screens (blocks) for continuation.

Figure 4 presents an example drawing. The data file is given in Listing 3 as screens 40 and 41. Figure 5 illustrates the derivation of the data file. The first five line segments are

Figure 4. An example drawing

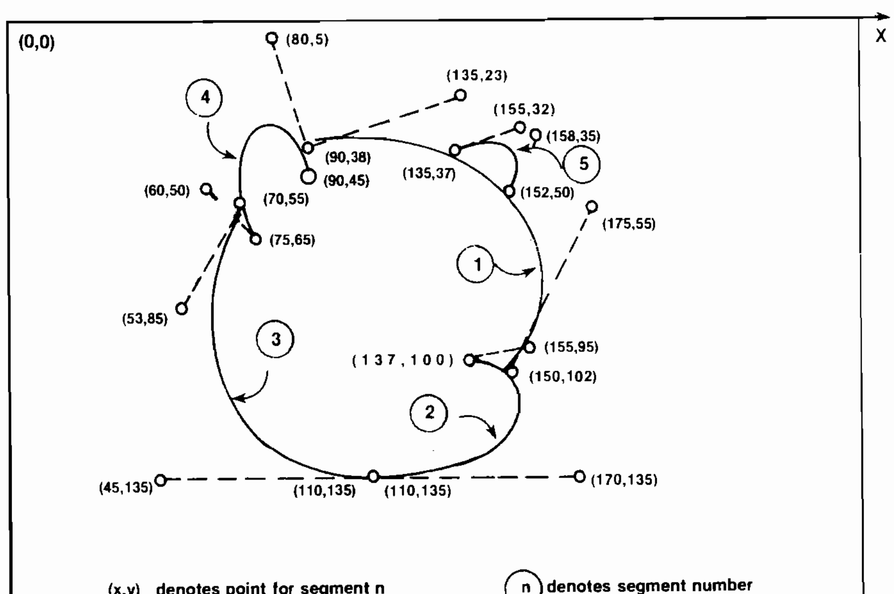


Figure 5. Illustration of data definition

shown with points P0 through P3 for each. To draw the figure defined by these data the command

40 FIGURE

would be entered at the terminal.

REFERENCES:

1. Rogers and Adams, Mathematical Elements for Computer Graphics, McGraw-Hill Book Co., 1976, pp. 139-144.

2. Monroe, A.J., FORTH Dimensions, Volume III, No. 6, p. 187.

Richard H. Turpin may be reached at 8226 Warbler Way, Indianapolis, IN 46223.

Listing 1. Bezier Curves in FORTH

```
        SCR # 42
( Bezier Curves - A FORTH Implementation                1/25/83)
( By R. H. Turpin
( Reference:  Rogers & Adams, "Mathematical Elements for       )
(    Computer Graphics,"  pp 139-144, McGraw-Hill, 1976.       )
( Define 4 points P0, P1, P2, and P3, each an X,Y pair.        )
( The Bezier curve is defined by the following equation,       )
( where 0<=T<=1:                                               )
(    P[T] = P0*[1-T]^3 + 3* P1*T*[1-T]^2                       )
(              + 3*P2*T^2*[1-T] + P3*T^3                       )
( where ^ denotes "raised to the power."                       )
( The above equation can be rewritten as:                      )
(    P[T] = T*[T*[T*P3'+P2']+P1']+P0']                         )
( where P0' = P0, P1' = 3*[P1-P0], P2' = 3*[P2-P1]-3*[P1-P0] )
( and P3' = P3-P0-3*[P1-P0].                                   )
FORTH DEFINITIONS
-->

        SCR # 43
( Bezier Curves cont'd )
( In the following code control points P0...P3 will be        )
( referenced as X,Y pairs in terms of their corresponding     )
( names.  For example, P2 will refer to X2,Y2.  To            )
( facilitate this the following operators are defined.      )
: P@  ( ADDRESS OF X --- X,Y )  DUP @ SWAP 2+ @ ;
: P!  ( X,Y,ADDRESS OF X --- )  ROT OVER ! 2+ ! ;
: P+  ( X0,Y0,X1,Y1 --- X0+X1,Y0+Y1 )  ROT + ROT ROT + SWAP ;
: P-  ( X0,Y0,X1,Y1 --- X0-X1,Y0-Y1 )
    ROT SWAP - ROT ROT - SWAP ;
: P*  ( X,Y,N --- NX,NY )  ROT OVER * ROT ROT * ;
: P*/  ( X,Y,M,N --- MX/N,MY/N )
    ROT >R OVER OVER >R >R */  ( M*X/N )
    R> R> R> SWAP */  ( M*Y/N ) ;
-->

        SCR # 44
( Bezier Curves cont'd )
( Point operators continued )
: PDUP  ( X,Y --- X,Y,X,Y )  OVER OVER ;
: PSWAP  ( X0,Y0,X1,Y1 --- X1,Y1,X0,Y0 )
    >R ROT ROT R> ROT ROT ;

( Storage for control points P0...P3 )
VARIABLE P0 2 ALLOT ( X0,Y0)    VARIABLE P1 2 ALLOT ( X1,Y1)
VARIABLE P2 2 ALLOT ( X2,Y2)    VARIABLE P3 2 ALLOT ( X3,Y3)
( Storage for modified points P0'...P3' )
VARIABLE P0' 2 ALLOT            VARIABLE P1' 2 ALLOT
VARIABLE P2' 2 ALLOT            VARIABLE P3' 2 ALLOT

VARIABLE N  ( Number of line segments in curve )
VARIABLE N.SC  3 N.SC !  ( Scale factor for N calculation )
-->
        SCR # 45
( Bezier Curves cont'd )
( Compute modified points P0'..P3' from control points P0..P3 )
: COMPUTE.P'  P1 P@ P0 P@ P- 3 P* ( 3[P1-P0] )  PDUP
        P2 P@ P1 P@ P- 3 P*  ( 3[P2-P1] )  PDUP
        P3 P@ P0 P@ P-  ( [P3-P0] )  PSWAP P-
        P3  P! PSWAP P- P2  P! P1  P!  P0 P@ P0' P! ;

( A few utilities for handling/defining control points )
( Define control points )
: STORE.POINTS  ( P0,P1,P2,P3 --- )
    P3 P! P2 P! P1 P! P0 P! ;
( Display control points at CRT )
: LIST.POINTS  CR P0 P@ . . P1 P@ . . P2 P@ . . P3 P@ . . CR ;
( Display control points on graphics screen )
: SHOW.POINTS  P0 P@ TDOT P1 P@ TDOT P2 P@ TDOT P3 P@ TDOT ;
  -->

        SCR # 46
( Bezier Curves cont'd )
( Move cursor to start of curve )
: MOVE.TO.P0  P0 P@ TMOVE ;
( Load modified control points onto stack )
: LOAD.POINTS  P0' P@ P1' P@ P2' P@ P3  P@ ;
( Compute point on curve given control points and            )
( parameter, T, where 0 <= T <= N.                           )
: P[T]  ( P0',P1',P2',P3',T --- P[T] )
    3 0 DO DUP >R N @ P*/ P+ R> LOOP DROP ;
( Compute the number of curve segments by finding the        )
( distance from P0 to the midpoint on the curve.             )
( The number of segments equals the larger of delta X or     )
( delta Y, with a minimum of 3 and a maximum of 50.          )
: FIX.N  LOAD.POINTS 2 N ! 1 P[T]  ( compute midpoint )
    P0 P@ P- ABS SWAP ABS MAX  ( select maximum from X or Y )
    N.SC @ / 3 MAX ( min of 3) 50 MIN ( max of 50) N ! ;  -->

        SCR # 47
( Bezier Curves cont'd )
( Draw a Bezier curve )
: CURVE  N @ 1+ 1 DO LOAD.POINTS I P[T] TDRAW LOOP ;

: BEZIER  COMPUTE.P'  FIX.N  MOVE.TO.P0  CURVE ;

( To use the above do the following:                         )
(     step 1 - define control points using STORE.POINTS      )
(     step 2 - execute BEZIER                                )
( The curve generated will pass through the two end          )
( points P0 and P3.  The curve will leave point P0 with      )
( direction defined by a line connecting P0 and P1, and      )
( will approach P3 with direction defined by a line          )
( connecting P2 and P3.  Control points P1 and P2 are        )
( not on the curve but they do help determine the            )
( shape of the curve.                             )  ;S
```

Listing 2. Bezier Application - Line Drawings

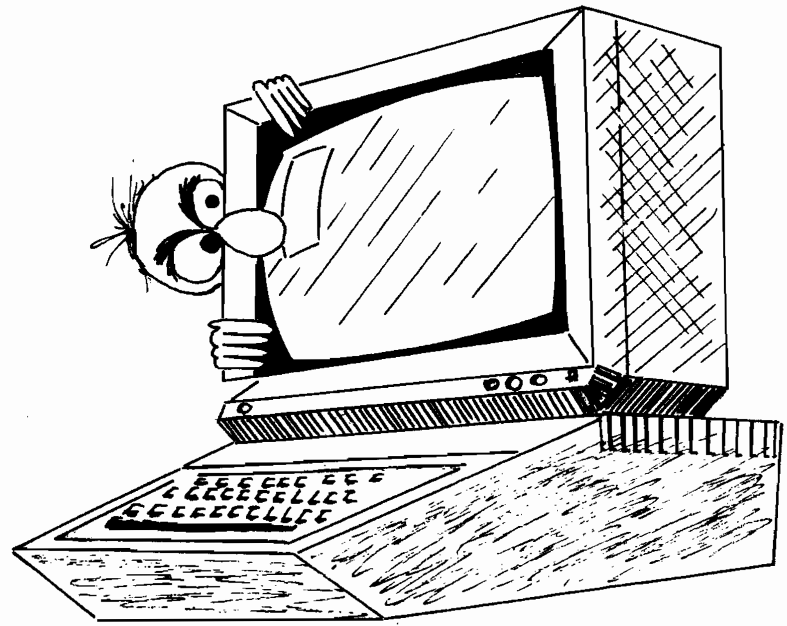
```
        SCR # 35
( Bezier Curve Application:  Line Drawings            1/4/83 )
( By R. H. Trupin
( Commands for generating line drawings:                     )
(     B - Draw a curve segment [Data:  P0,P1,P2,P3]          )
(     C - Draw a circle [Data:  X,Y,RADIUS]                  )
(     D - Plot a dot [Data:  X,Y]                            )
(     E - End of figure [Data: none]                         )
(     H - Set color [Data:  COLOR]                           )
(     L - Draw a line [Data:  X0,Y0,X1,Y1]                   )
(     N - Read another disk block [Data:  BLK NO.]           )
(     R - Draw a rectangle [Data:  BASE,HEIGHT]              )
(     X - Clear screen [Data: none]                          )
42 LOAD  ( Load Bezier and graphics) 60 LOAD ( Load CASE )
( Constants for command reference )          66 CONSTANT =B
67 CONSTANT =C 68 CONSTANT =D 69 CONSTANT =E 72 CONSTANT =H
76 CONSTANT =L 78 CONSTANT =N 82 CONSTANT =R 88 CONSTANT =X-->
```

```
    SCR # 36
( Line Drawings cont d                        RHT:1/4/83)
( Words to service commands )
: DATA  ( Read word from data; convert to number on stack. )
   32 WORD NUMBER DROP  ;
( Load a block of data from disk; print header )
: READ.BLK  ( BLOCK NO. --- )  DUP SCR ! BLOCK DROP
   SCR @ BLK ! 0 >IN ! 41 WORD COUNT TYPE ." )" CR ;
( Command B:  Draw curve using Bezier function )
: DRAW.CURVE
   8 0 DO DATA LOOP ( read 4 X,Y pairs, points P0,P1,P2,P3 )
   STORE.POINTS BEZIER ( Draw curve )   ;
( Command C:  Draw a circle )
: DRAW.CIRCLE  DATA DATA DATA TCIRCLE  ;
( Command D:  Plot a dot )
: PUT.DOT    DATA DATA TCOT    ;  -->

    SCR # 37
( Line Drawings cont'd                        RHT:1/4/83)
( Command E:  End of drawing )
: END.TASK  ." End of figure."  ABORT ;
( Command H:  Set plotting color )
: SET.COLOR  DATA COLOR ! ;
( Command L:  Draw a line )
: DRAW.LINE
   4 0 DO DATA LOOP ( get X0,Y0,X1,Y1 )
   TMOVE TDRAW  ( draw line )  ;
( Command N:  Read next block of data )
: NEXT.DATA   DATA READ.BLK ;
( Command R:  Draw a rectangle )
: DRAW.RECTANGLE  DATA DATA TRECT ;
-->

    SCR # 38
( Line Drawings cont'd                        RHT:1/4/83)
( Command X:  Clear the screen )
: CLR.SCREEN  CLEAR   ;
( Command execution )
: EXEC.COMMAND  ( COMMAND ---  )
   CASE  =B  OF  DRAW.CURVE      ENDOF
         =C  OF  DRAW.CIRCLE     ENDOF
         =D  OF  PUT.DOT         ENDOF
         =E  OF  END.TASK        ENDOF
         =H  OF  SET.COLOR       ENDOF
         =L  OF  DRAW.LINE       ENDOF
         =N  OF  NEXT.DATA       ENDOF
         =R  OF  DRAW.RECTANGLE  ENDOF
         =X  OF  CLR.SCREEN      ENDOF
         ." Bad data in block! " BLK @ . CR ABORT
   ENDCASE ;    -->

    SCR # 39
( Line Drawings cont'd                        RHT:1/4/83)
( Get command from data;  leave on stack )
: READ.COMMAND  ( COMMAND ADDR. --- COMMAND )
   32 WORD COUNT DROP @ 255 AND ;

( Word to draw figure using data stored on disk )
: FIGURE  ( BLOCK NO. --- )
   READ.BLK  ( Load in figure data file )
   BEGIN  ( Draw until end of figure command )
      ?TERMINAL  IF ABORT THEN  ( Test for operator abort )
      READ.COMMAND
      EXEC.COMMAND      0  ( False flag to continue )
   UNTIL ;
;S
```

Listing 3. Example Line Drawing Data File

```
    SCR # 40
( LINE DRAWING DATA:  PORKY PIG ) X  H 1
B 90 38 135 23 175 55 150 102  B 137 100 155 95 170 135 110 135
B 110 135 45 135 53 85 70 55
B 75 65 60 50 80 5 90 45
B 135 37 155 32 158 35 152 50
B 85 105 95 100 103 97 108 100
B 105 103 109 100 110 100 113 100
B 111 108 117 85 140 85 133 108  L 111 108 133 108
B 133 108 125 130 103 130 110 113  L 110 113 130 113
B 110 119 112 117 113 117 120 120
B 118 118 122 117 123 117 127 119
B 100 98 115 65 122 70 113 100 B 108 97 115 85 117 85 113 100
B 133 95 145 70 148 75 142 98  B 135 98 143 88 145 88 142 98
H 15  D 40 15  R 165 135        N 41

    SCR # 41
( PORKY PIG CONTINUED )  H 6
C 55 160 5  L 50 155 50 170
C 70 165 5
L 80 160 80 170 B 80 163 84 160 86 160 90 162
L 95 155 95 170  L 95 163 103 157  L 98 162 103 170
B 110 160 110 175 120 175 120 160
B 120 160 120 180 110 180 110 172
C 150 160 5  L 145 155 145 170
L 160 163 160 170  C 160 157 2
C 170 165 5  B 175 160 175 180 165 180 165 172  E
```

PEEKing Tom
Playing with BASIC's Internals



by Mark Johansen

A few simple techniques to help find how and where things are done within almost any system

One thing that I enjoy about microcomputers is that no one minds if you mess with the system internals: unlike a large mainframe, where any disruption to the system may affect hundreds of people and cost the company a lot of money, on a micro the only one you can really hurt is yourself, and if you do it's generally your own fault. In fact, most of the micros on the market today come with a BASIC interpreter that includes a PEEK function and a POKE statement. It is hard to think of a feature which could be handier to someone interested in seeing just how the system works and occasionally taking advantage of such internals.

I am using an IBM PC at work and, much to my disappointment, none of the manuals that I have for it say much of anything about where the system keeps its variables or how BASIC programs or data are actually stored in memory. However, I managed to devise a few simple techniques to help me find how things were done. While I will use the IBM PC as my reference point, the ideas I discuss here should be applicable to almost any micro having PEEKs and POKEs. For this reason, I will avoid using any statements or formats in my sample programs that are not likely to be found in almost any version of BASIC (such as NEXT without a variable, multiple statements per line, the PC's DEF SEG, etc.).

[Ed. Note: These mini-programs and techniques are really very easy to adapt to other micros. I used them on the Apple, Commodore and running 6809 Flex BASIC.]

Finding A Program

A logical place to start is with finding our program. It is extremely unlikely that the operating system will actually place our program at the beginning of memory; it probably has data it needs for itself there. But what we can do is write a short program that will look through memory searching for itself. How will it recognize itself when it finds it? If we knew exactly how the system stored our program in memory this would be no problem: we would just look for a string of bytes which matches the first few bytes of our program. But even if we do not know this, there is one thing we can be reasonably confident will be recognizable, namely, text such as in a PRINT statement.

Consider the following:

Program 1

```
10 PRINT "FINDING MYSELF AT:"
20 FOR S=0 TO 10000
30 C1$=CHR$(PEEK(S))
40 IF C1$<>"F" THEN 90
50 C2$=CHR$(PEEK(S+1))
60 IF C2$<>"I" THEN 90
70 PRINT "FOUND MYSELF AT";S
90 NEXT S
100 END
```

This program will search through memory looking for every occurrence of the letters "FI", the first two letters of the printed text string, and printing out the address of each occurrence. (The 10000 in the FOR statement is a reasonable amount of memory to search and is likely to contain your program. If you cannot get a hit and you have more than 10,000 bytes of memory, you might increase this number. If the program keeps running for a long time after it has found itself, you may just break it and not worry about looking any further.) If there is more than one fine, there is probably just another place in memory that contains those two characters by coincidence. We simply change the program to check for more letters before reporting a success. When we have narrowed it down to one we have found our program. Of course,the real start will be a few bytes before this, as the line number, the keyword PRINT, and possibly some control information must precede the text string.

You might try putting in programs of different lengths (just tack some useless lines on the end), adding more variables, starting with different things on the screen, running with and without any memory expansions you might have and so on, to see what, if anything, changes your program's location.

Finding the Program Start Address

While we may find it amusing to see where our program begins, it is imperative that the operating system be able to find it if it hopes to do anything meaningful when we type "RUN". It is, therefore, very likely that it keeps track somewhere of what our program's actual start address is. If we knew where this value was stored, we could save ourselves the trouble of having to look for our program every time and simply pick the number up from the same place that the operating system gets it.

How might we find it? We could cheat and look in the manual, but then it's always possible that the manual doesn't say. A more interesting approach would be to search for our program as above, finding the address at which it begins, and then search through memory for a location containing this address. This location is likely to be the system's start-of-program variable.

There are two points that must be dealt with. First, Program 1 does not find us the actual start address, but rather something a little ways after that. We can get around this by simply looking for an address which is **slightly before** the address we found for our PRINT text. Second, we must consider how the computer actually stores addresses internally. The microprocessors that I am familiar with (8088, 8080/Z80, 6502) all store addresses in two bytes with the first byte containing the **last** eight bits of the number and the second byte containing the **first** eight bits. We can pick up such a number in a BASIC program by coding **PEEK(A) + 256*PEEK(A + 1)** (where A contains the address of the first byte of the number). It is possible that your computer stores addresses differently, in which case you would have to adjust line 220 in the program below. If you don't know how your computer stores addresses, just try it as I have it and see what happens.

[Ed. Note: The 6800 and 6809 are exceptions. They store the address with the high eight bits in the first byte and the low eight bits in the second byte.]

Program 2

```
10 PRINT "FINDING MYSELF AT:"
20 FOR S=0 TO 10000
30 C1$=CHR$(PEEK(S))
40 IF C1$<>"F" THEN 90
50 C2$=CHR$(PEEK(S+1))
60 IF C2$<>"I" THEN 90
70 PRINT "FOUND MYSELF AT";S
80 GOTO 200
90 NEXT S
100 END
200 REM LOOK FOR START ADDRESS
210 PRINT
220 FOR P=0 TO 10000
230 A=PEEK(P)+256*PEEK(P+1)
240 IF A<S-10 OR A>S THEN 260
250 PRINT P;"CONTAINS";A
260 NEXT P
270 END
```

If you get several "hits", see which one looks most likely and play with it a bit. You might try doing PEEKs at the addresses of these variables and seeing what's there. Running this on the PC gave me two hits: one at location 48 and another at location 862. A little examination showed that location 48 contains the start address. (We'll get back to what is in 862 later.)

Dumping a Program

Now that we have found where the program is kept, we might try dumping it out to see how it really is stored internally. All we have to do is start at the address we found above and dump the next hundred bytes or so. To help us discern what each byte is as it is dumped, we can print it as both a number and a character (the first we get directly from the PEEK statement, the second we can get with the CHR$ function). On the IBM PC the program start address is stored at locations 48-49, so I will use that location in the program below. You would, of course, have to substitute the appropriate value for your computer here.

[Ed. Note: The Commodore 64 uses locations 43 and 44 – but you know that from Program 2. The examples have been modified to use this address. To change to another micro, simply re-define X and Y in line 10. The values would be 48 and 49 for the IBM PC.]

Program 3:

```
10 X=43:Y=44
20 S=PEEK(X)+256*PEEK(Y)
30 PRINT "STARTING ADDRESS =";S
100 REM  START DUMPING
110 FOR A=S TO S+100
120 C=PEEK(A)
130 PRINT C;"(";CHR$(C);")";
140 NEXT A
150 END
```

The output of this program will not look much like a BASIC listing, and not just because of those numbers stuck in there. Though variable names and the text within PRINT and REM statements should be recognizable, that may be just about it.

There are several ways in which BASIC's commonly code their statements. Rather than store keywords such as PRINT or CHR$ as several characters, most BASIC interpreters use codes from the character set that are not used for any of the regular letters, numbers, or symbols. There one-byte codes are referred to as "tokens". You should be able to quickly identify what token your computer uses for PRINT by looking at what comes before the recognizable text; to figure out other tokens you should study the context in a similar fashion and find something that looks consistent. We'll get back to looking at tokens in a moment.

Some BASIC's store numbers as strings of digits: if this is the case with your computer they should be easily recognizable. Others store numbers in binary, which will force you to do some conversions if you want to be able to read what is there.

A particularly interesting thing to look for is what your BASIC puts at the front and end of each line. The BASIC's I have seen all use a zero byte to mark end-of-line, and have the line number in binary at the front. Most also put a link in front of the line number: a two-byte field which holds the address of the start of the next line.

Once you have an idea of how your programs are stored, you might try making the dump more comprehensible. For example, you could determine how to find where a new statement begins and print your dump of each statement starting on a new line. This makes it much more readable.

[Ed. Note: I took Mark's suggestion and wrote the following program that dumps lines of BASIC. If the value is a printable ASCII character (range 32 to 127), then it is printed as a character. Otherwise, it is output as a decimal value in parentheses. These values would be the tokens. The C64 version uses some of the color and reverse features to make a more readable display. X and Y point to the program start found in the earlier programs.]

Tokens

It might be entertaining to try to get a complete list of all of our BASIC's tokens. We could do this by writing a program that includes every single BASIC keyword and then dumping it out, but there is an easier way: we can let a program construct every possible token and then LIST them to tell us what they are. We begin the program with a dummy line consisting of only one token: REM. When we RUN the program this will, of course, do nothing, but the program will then go back and change the REM token, tell us what it changed it to, and then LIST that line to show us what keyword that token corresponds to.

[Ed. Note: Mark's IBM PC version was changed to use the C64 references.]

Program 4

```
10 REM
20 X=43:Y=44
30 S=PEEK(X)+256*PEEK(Y)
40 T=S+4
200 REM CHANGE TOKEN AND LIST
210 V=129
220 POKE T,V
230 PRINT V
240 LIST 10
250 V=V+1
260 GOTO 220
```

The program should print out the number 129 followed by a listing of line 10: the number 10 and then the keyword corresponding to the token 129. For the PC this comes out

129
10 END

Unfortunately, on the PC at least, this is all it does.

[Ed. Note: This works for the Apple II, but not the C64.]

The problem is that the program ends once it finishes the list statement rather than going on to execute the next instruction and looping around. If you do not have this problem, count your blessings. For those of you who do, there are two ways we might overcome this. One way is to simply type GOTO 250 after each token is listed to get the next one. (Note: if you type GOTO 250 the variables will probably be left as they were when the program finished; if you type RUN 250 they will more than likely be cleared and you will end up poking the number one into location zero.) Another possibility is to put several dummy REM statements at the front of the program, poke token values into each of them, and then LIST. We will have to know where to do the POKEs, of course. Just as we knew where to put the first one by adding the number of bytes preceding the first token on a line to the program start address, we can find the others by adding the number of bytes in each line to the place where we did the last poke and looping along. For the IBM PC, the length of each line is 2 for the link plus 2 for the line number plus 1 for the REM token plus 1 for the end-of-line marker (a null), which gives 6. This number is used in line 230.

Program 4b

```
10 REM
20 REM
30 REM
40 REM
50 REM
60 REM
70 REM
80 REM
90 REM
100 X=43:Y=44
110 S=PEEK(X)+256*PEEK(Y)
120 T=S+4
200 REM CHANGE TOKEN AND LIST
210 FOR V=129 TO 137
220 POKE T,V
230 T=T+6
240 NEXT V
250 PRINT "129-137:"
260 LIST 10-90
```

BASIC Dump

```
10 X=43:Y=44
20 S=PEEK(X)+256*PEEK(Y)
30 PRINT "STARTING ADDRESS =";S
100 REM  START DUMPING
110 FOR I=1 TO 10
120 P=PEEK(S)+256*PEEK(S+1):PRINT P;
130 Q=PEEK(S+3)*100+PEEK(S+2):PRINT Q;
140 S=S+3
150 S=S+1:R=PEEK(S)
160 IF R>31 AND R<128 THEN PRINT CHR$(R);:GOTO 150
170 IF R<>0 THEN PRINT "(";R;")";:GOTO 150
180 PRINT:S=S+1
190 NEXT I
200 END
```

C64 BASIC Dump

```
10 X=43:Y=44
20 S=PEEK(X)+256*PEEK(Y)
30 PRINT "STARTING ADDRESS =";S
100 REM  START DUMPING
110 FOR I=1 TO 10
120 P=PEEK(S)+256*PEEK(S+1):PRINT "{RED}"P;
130 Q=PEEK(S+3)*100+PEEK(S+2):PRINT "{BLUE}"Q"{BLACK}";
140 S=S+3
150 S=S+1:R=PEEK(S)
160 IF R>31 AND R<128 THEN PRINT CHR$(R);:GOTO 150
170 IF R<>0 THEN PRINT "{RVS}"R"{RVSOFF}";:GOTO 150
180 PRINT:S=S+1
190 NEXT I
200 END
```

By the way, if you try to run this program a second time, it is not likely to work, as you have changed the first few lines, most likely into syntax errors. You will have to re-enter the dummy REMs before you try to re-run (or you could cheat and say RUN 100).

This program will tell us what the tokens 129 through 137 correspond to. If we want to get the full list, we could either add more dummy REMs and enlarge the loop, or we could simply run it several times, changing the boundaries of the loop each run.

If you are wondering why I chose the number 129 to start with, it is because that begins the second half of the character set (1/2 * 256 possible character codes in an 8-bit byte + 1 = 129) and therefore seems a likely place to include tokens. The first half most likely includes your alphabet, digits, and other printable symbols. You might try running Program 4b using V values less than 129 and see what you get.

[Ed. Note: I am not sure why Mark did not start at 128. This is a valid token on the Apple, C64 and Flex BASIC. It may be different on the IBM PC]

One final problem you may have to deal with: on the IBM PC, some of the tokens really take two bytes, a 255 followed by something else. Also, some tokens mean that what follows is a number of some kind (one byte integer, two byte integer, GOTO address, etc), which therefore logically requires that something follow before the end of the line. In there cases, the above program will result in the beginning of the next line being "eaten" to satisfy the requirement for extra bytes, and from there on everything is a mess. If you run into this situation, simply put a few extra characters on each dummy statement. (And remember to change line 230 to keep your POKEs landing in the right place!)

Another System Variable: The DATA Pointer

The problem that originally led me to work on examining system internals was a BASIC program I was working on that included many, many DATA statements. My program logic was able to spot certain types of errors in the data, and I wanted it to print messages saying what was wrong and where it had found the problem. The best way to say "where" would be to give the line number. Thus, I wanted to find where the system kept track of what data line it is looking at.

For the IBM PC, surprise! We get location 862, our old friend from looking for start of program. When we were not using any DATA statements, it pointed to one byte before the start of the program. Evidently that is where it begins life before the first READ is executed.

Of course, what I really wanted for my original program was the **line number,** not the address, but this can be found by simply tracing through the links at the start of each line. When we find the first line with a link which contains an address larger than the data pointer, it follows that the pointer must be aiming somewhere in the current line. We then pick up the line number and we've got it.

Final Words

Other interesting things to look for are your memory-mapped video, your own BASIC variables, and the system clock. I could go on demonstrating exactly how I found a couple of other system variables, but I think the technique should now be clear: whatever it is you may be looking for,do something which will set it equal to a value you can calculate or predict yourself, then search through memory for a location containing that value. If you find more than one, try to set up a **different** predictable value and do it again. If you do not find any, either you have made a mistake somewhere or the system does not keep any such value.

Do not be fooled by coincidences! When I was first looking for my DATA statements, I looked for a location containing the line number rather than the address. The PC does not keep any such value, but I found one anyway. No matter what line I had my READs reaching, location 823 would always contain the line number, but when I tried to use it in a program it did not work. It turned out that what I was finding was some work area in which the system had placed my line number, probably preparatory to doing the IF test.

In general, it is wise to try things out in a small program before spending a lot of time working from a faulty assumption. Which statement could no doubt be applied in many other contexts.

The technique we use is very similar to the way in which we found the pointer to the start of the program. We set up a DATA statement and execute a READ, so that the data pointer should now be pointing into our DATA line. We than search through memory for any location containing a value near the beginning of the program. We could determine the exact value to look for if we would take the time to dump the program and see just where everything falls, but we would still have to worry about whether the pointer would be at the last digit of the number just read, the comma, or the first digit of the next number, so we may just as well be lazy, put the DATA statement near the front, and figure that those lines can't take more than 25 bytes or so. Thus:

Program 5

```
10 READ X
20 DATA 34,24,36
100 S=PEEK(48)+256*PEEK(49)
110 FOR P=0 TO S
120 S1=PEEK(P)+256*PEEK(P+1)
130 IF S1<=S OF S1>S+25 THEN 150
140 PRINT P;"CONTAINS";S1
150 NEXT P
1670 END
```

[Ed. Note: Just for fun, I wrote the following program to print all of the tokens on the C64. S is the address of the first token, determined by 'fooling around' with Mark's programs. A sample of the printout is included.]

```
10 S=41118: REM FROM FIND TOKEN FOR C64
200 PRINT "ADDRESS   NUMBER      TOKEN"
210 FOR X=128 TO 255
220 IF PEEK(S)=0 THEN X=255:GOTO 260
230 PRINT S,X,;
240 IF PEEK(S)<128 THEN PRINT
    CHR$(PEEK(S));:S=S+1:GOTO 240
250 PRINT CHR$(PEEK(S)-128):S=S+1
260 NEXT X
270 PRINT:PRINT "END OF TOKENS"
280 END
```

```
RUN OF PRINT TOKENS FOR COMMODORE 64

ADDRESS   NUMBER     TOKEN
 41118    128        END
 41121    129        FOR
 41124    130        NEXT
           .          .
           .          .
           .          .
 41361    201        RIGHT$
 41367    202        MID$
 41371    203        GO

END OF TOKENS
```

MICRO™ Interface Clinic

by Ralph Tenny

The last column dealt with the A/D and D/A converters of the "conventional" kind. That is, they used resistor ladders to generate DC voltages proportional to a binary word. There are numerous other methods to make A/D and D/A conversions, and we will discuss some of them this time. A few A/D techniques offer unique advantages for making special measurements, and some converter ICs offer special advantages for low cost conversion.

The Voltage-to-Frequency (V/F) converter produces an output frequency proportional to an input current or voltage. A typical low-cost unit easily measures the range from .01 volts to 10 volts with a resolution of .001 V (1 mv) and linearity of 1 mv. That measurement range yields frequencies between 10 Hz and 10,000 Hz. Note that this corresponds to a binary resolution of 12 bits at 10 volts. That resolution in most other kinds of technology would cost four times as much. Now for the major disadvantage - each measurement takes a full second!

There are tricks to compensate for the slow readings, if your need to. For example, if you expect to measure a voltage close to full scale, you can sample for a shorter period. With a .001 second sample, 10 volts input gives 100 counts full scale. The catch is that the resolution, linearity and accuracy all degrade in proportion.

If you want to measure small voltages, you can measure the period (time between two successive pulses) to quite good resolution, then compute the frequency. With the original calibration of 1 Hz/mv, the period for 10 mv input would be 100 msec. If you use a peripheral counter in the computer, this measurement can be made to 1 usec resolution, or 100,000 counts. Obviously, this method runs out of resolution at 10 volts input - it gives 100 counts full scale. Another caution with period measurement is that the input voltage must be heavily filtered or very steady to avoid period-to-period variations. In general, if an experiment can be set up with a narrow range of input voltage, choose either frequency or period measurements for the most acceptable results.

Some A/D converter technologies are a mix of analog and digital techniques. One of the first technologies to be developed was single- and multi-slope integrating A/D converters. Figure 1 shows the basic premise of single-slope integrator conversion. A linear ramp is compared against a DC voltage. The ramp is started at the same time as an oscillator; a digital counter is allowed to count the number of oscillator cycles which occur until the ramp rises to equal the input voltage. The count in

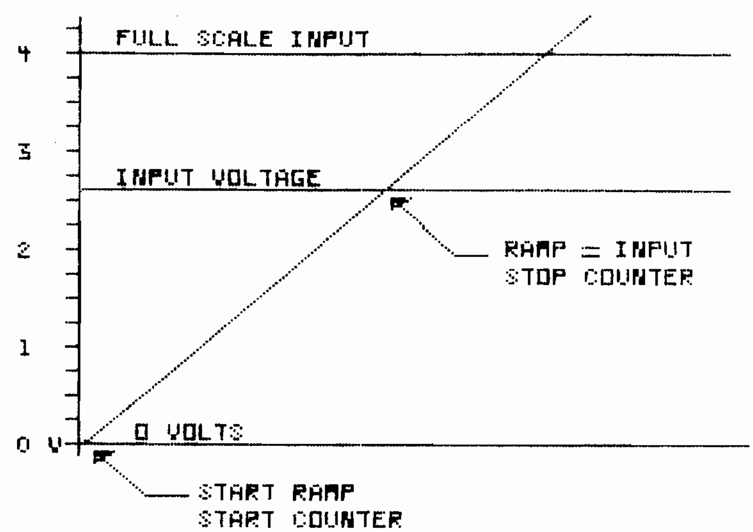


Figure 1. Operating scheme for a single-slope A/D converter.

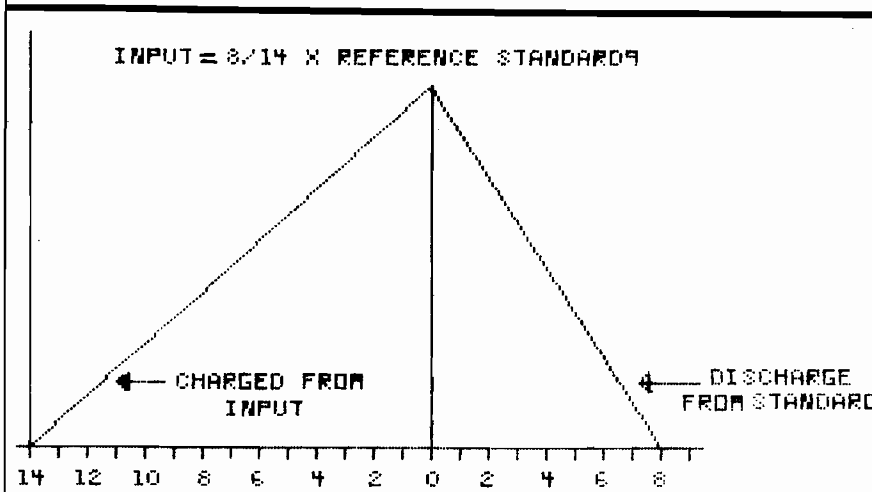
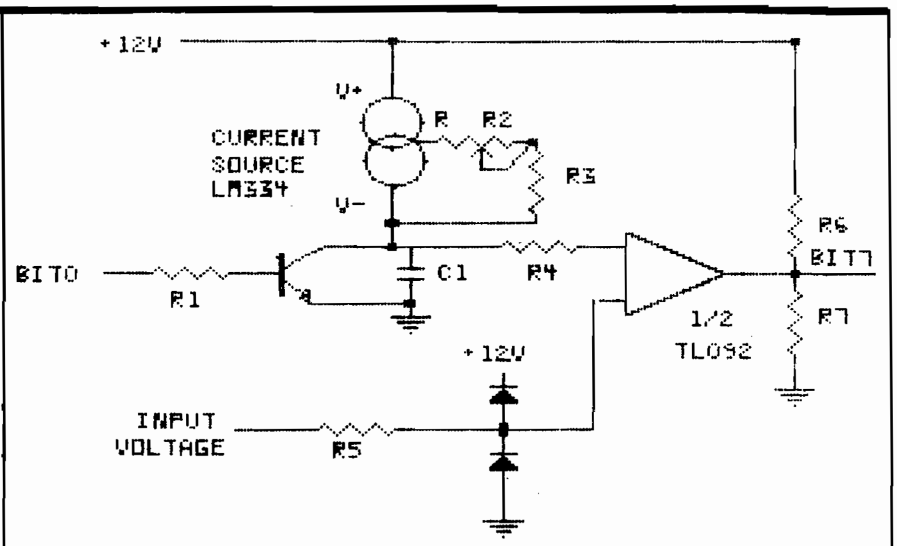


Figure 2. Operting concept of a dual-slope integrating A/D converter. A capacitor is charged by the input voltage for a fixed time, then discharged by a reference source. The input voltage is determined by a ratio of charge to discharge times.

the counter then is proportional to the DC voltage input. This is quite similar to the converter proposed in Figure 2 of the previous column (MICRO #69, February 1984); the difference is that we now are using a free-running analog ramp instead of a digital stair-step ramp. Also, the count is now in a digital counter instead of a memory location.

Figure 2 shows the concept of dual-slope integration used for A/D conversion. During the first half of the operating cycle, the unknown voltage is allowed to charge a capacitor for a known period of time. In the second period, this capacitor is discharged by a reference source. The discharge time is compared to the standard sample time, and the resulting number is proportional to the input voltage. If it takes half as long to discharge the capacitor as it takes to charge it, the input is one-half of full scale.

When evaluating A/D converters, several factors need to be considered. In past discussions we have studied trade-offs between accuracy and resolution. Cost usually is a factor, but the one parameter which varies most widely is conversion time. The previous column discussed the successive



Parts List

R1,R4,R5,R6	**6.8 ohm ¼ watt resistor**
R2	**2K ohm pot**
R3	**820 ohm, ¼ watt resistor**
R7	**5.6K ohm ¼ watt resistor**
Current source LM334 (National Semiconductor)	
Op Amp	**TL092 (Ratio Shack 276-1746)**
Diodes	**1N4148 or 1N914**
Transistor	**General purpose silicon NPN**
C1	**.01 mF capacitor**

Figure 3. Simple single-slope integrating A/D conveter. It requires only two I/O lines.

approximation converter. This technology is the fastest of the lower cost technologies. A typical converter will complete a conversion in about 50 usec. We just noted that F/V converters require close to a second to make a full-resolution measurement. Integrating converters can typically make a conversion in 1/10 to 1/100 second.

Let's look at some real hardware! Beginning with this column, some of our interfacing experiments will be done on the Commodore 64. There are several plug-in peripherals available for the C64. If you have one, you can make the necessary translation from the pinout I will describe to your own setup.

If you plan to do any experimentation with your C64, you should purchase the *Commodore 64 Programmer's Reference Manual*. This book is about an inch think and is jammed with crucial assembly-language and hardware information, in addition to a lot of information on BASIC programming. In the I/O section you can find pinout information on the User port.

I made a simple adapter to give easy access to the C64 User port, and will use it for all simple interfacing projects. The major hurdle to building your own adapter is to find a connector which fits on the User port. The User port is a card edge protruding from the C64 case at the left rear. You need a PC card edge connector with double readout (12/24) pins on .156" spacing. Either solder eyelet or wire-wrap type pins are acceptable. The part number for one such socket manufactured by TRW is 251-12-50-171. I cannot find this connector type in the mail-order catalogs I have, but it should be available from electronic parts distributors. If you don't mind a little handwork, you can cut up a Radio Shack #276-1551.

To modify the Radio Shack connector, use a fine-tooth saw to cut off both lugs. Preserve one end of the connector body intact, and count over 13 pin positions. Saw the connector off in the middle of the 13th pin position. Note that the User port connector has slots between pins 1-2 and 10-11. Cut a small piece of 1/64" plastic or fiberboard to fit in between the pins of the connector at those positions. With these keys in place, you won't be able to push the connector on unless it is properly lined up. **NEVER** plug or unplug any C64 connectors with power

Listing 1

```
                    ; THIS PROGRAM EXERCISES A SINGLE-SLOPE
                    ; A/D CONVERTOR WHICH IS DRIVEN BY THE
                    ; C64 USER PORT. IT USES TWO I/O LINES
                    ; AND BOTH TIMERS TO CONVERT AN ANALOG
                    ; INPUT TO AN EQUIVALENT COUNT STORED IN
                    ; A ZERO PAGE BUFFER.
                    ;
                    ; EQUATES
                    ;
                    ; PAGE ZERO
007F                 DELYLO   EQU $7F
007E                 DELYHI   EQU $7E
007D                 CNTLO    EQU $7D
007C                 CNTHI    EQU $7C
                    ;
                    ; CIA CONTROLLER
DD01                 BPORT    EQU $DD01
DD03                 BDDR     EQU $DD03
DD04                 TMRALO   EQU $DD04
DD05                 TMRAHI   EQU $DD05
DD06                 TMRBLO   EQU $DD06
DD07                 TMRBHI   EQU $DD07
DD0E                 TMRACR   EQU $DD0E
DD0F                 TMRBCR   EQU $DD0F
                    ;
C000                          ORG $C000
                    ;
C000 78              START    SEI          ; STOP INTERRUPTS
C001 A0 04                    LDY #$04     ; LSB FOR TIME A
C003 8C 04 DD                 STY TMRALO
C006 A2 01                    LDX #$01     ; INIT PORT LINES
C008 8E 03 DD                 STX BDDR     ; BIT 0 OUTPUT
C00B A2 FF                    LDX #$FF     ; INITIAL TIMER B VALUES
C00D A0 41                    LDY #$41     ; TIMER B MODE
C00F A9 00                    LDA #00      ; START RAMP, STOP TIMER
C011 8D 05 DD                 STA TMRAHI   ; FINISH SETTING TIMER A
C014 8E 06 DD                 STX TMRBLO   ; INIT TIMER B COUNTS
C017 8E 07 DD                 STX TMRBHI
C01A 85 7F                    STA DELYLO   ; LOAD DELAY
C01C 84 7E                    STY DELYHI
C01E 8D 05 DD                 STA TMRAHI   ; COMPLETE INIT OF TIMER A
C021 8C 0F DD                 STY TMRBCR   ; START TIMER B
C024 A0 08                    LDY #08      ; MORE FOR TIMER A
C026 8D 01 DD                 STA BPORT    ; START RAMP
C029 8C 0E DD                 STY TMRACR   ; START TIMER A
                    ;
C02C 2C 01 DD        LOOP     BIT BPORT    ; TEST FOR RAMP EQUAL TO INPUT
C02F 10 FB                    BPL LOOP     ; SPIN UNTIL DONE
C031 8D 0E DD                 STA TMRACR   ; KILL TIMER B CLOCK
C034 8D 0F DD                 STA TMRBCR   ; ALSO TIMER B
C037 A9 01                    LDA #01      ; TURN OFF RAMP
C039 8D 01 DD                 STA BPORT
C03C AD 06 DD                 LDA TMRBLO   ; READ ACCUMULATED COUNT
C03F AE 07 DD                 LDX TMRBHI
C042 85 7D                    STA CNTLO    ; SAVE COUNTS
C044 86 7C                    STX CNTHI
C046 A0 00                    LDY #00      ; DISPLAY COLOR CODE
C048 A2 00                    LDX #00      ; SET INDEX POINTER
C04A A5 7C                    LDA CNTHI    ; GET DATA
C04C 20 5F C0                 JSR OUTPUT   ; SHOW IT
C04F A5 7D                    LDA CNTLO    ; NEXT DATA
```

(Continued on next page)

on!

After making the keyways fit properly, I bent each connector pin toward the pin directly opposite until the tips of the pins were about 1/16" apart. Now, any breadboarding circuit card (such as Radio Shack #276-152) will slide between the pins. Be sure to align the board in the connector so it does not overlap the cassette port. Solder each connector pin to an edge pin on the board and clean off all rosin residue completely. The final step is to put a 24 pin socket on the board and connect it to the socket so that the User port lines can be extended with a DOP jumper such as the Jameco DJ24-1-24. With this accessory, you can swap out any number of special boards. It is helpful if you use a specific order in making the connections between the socket and plug. The pinout I used was:

PORT PIN	SOCKET	FUNCTION
1	1	Ground
2	2	+5 VDC
3	3	RESET
4	4	CNT1
5	5	SP1
6	6	CNT2
7	7	SP2
8	8	PC2
9	9	ATN
10	10	9 VAC
11	11	9 VAC
12	12	Ground
A	24	Ground
B	23	FLAG2
C	22	PB0
D	21	PB1
E	20	PB2
F	19	PB3
H	18	PB4
J	17	PB5
K	16	PB6
L	15	PB7
M	14	PA2
N	13	Ground

Note that this particular pinout puts a circuit common (ground) connection at each corner of the 24 pin socket and gives redundant ground connections for better noise control. Also, the 9 port pins are grouped together in a logical order. The CNT, SP, FLAG and PC lines are special functions of the 6526 Complex Adapter Interface (CIA) and will be dealt with in a later column.

```
C051 20 5F C0           JSR OUTPUT
                 ;
C054 C6 7F       DELAY1 DEC DELYLO   ; .8 SECOND DELAY
C056 D0 FC              BNE DELAY1
C058 C6 7E              DEC DELYHI
C05A D0 F8              BNE DELAY1
C05C 4C 00 C0           JMP START    ; LOOP FOREVER
                 ;
C05F 48          OUTPUT PHA          ;SAVE DATA
C060 4A                 LSR          ; GET HI NIBBLE
C061 4A                 LSR
C062 4A                 LSR
C063 4A                 LSR
C064 20 74 C0           JSR CONVRT   ; MAKE DISPLAYABLE CHARACTER
C067 20 81 C0           JSR DISPLY   ; SHOW IT
C06A 68                 PLA          ; GET DATA
C06B 29 0F              AND #$0F     ; MASK TO LO NIBBLE
C06D 20 74 C0           JSR CONVRT
C070 20 81 C0           JSR DISPLY
C073 60                 RTS
                 ;
C074 C9 0A       CONVRT CMP #$0A     ; ALPHA OR NUMERIC?
C076 90 04              BCC NUMBER   ; 0 - 9
C078 38                 SEC          ; A - F
C079 E9 09              SBC #09      ; MAKE IT C64 SCREEN CODE
C07B 60          EXIT   RTS
                 ;
C07C 18          NUMBER CLC          ; CONVERT TO ASCII
C07D 69 30              ADC #$30
C07F D0 FA              BNE EXIT
                 ;
C081 9D 70 07    DISPLY STA $0770,X  ; PUT IN SCREEN BUFFER
C084 98                 TYA          ; SET CHARACTER COLOR
C085 9D 70 DB           STA $DB70,X  ; PUT IN COLOR RAM
C088 E8                 INX          ; BUMP INDEX
C089 60                 RTS
C08A 00                 BRK
                 ;
C08B                    END
```

You will note from the programming example below that the CIA port pins are easier to program than PIA lines previously discussed in this column. Also, the schematics shown below skip specific pinout details by showing direct connection between the port pins and the external circuit.

Figure 3 shows a rudimentary single-slope integrating converter. It isn't very accurate (+ or - 2% linearity between .5 volts and 5 volts) but it works well enough for experimentation. One resistor is marked (*); it is needed to restrict input level to the C64. If the circuit is used with CoCo, it should be removed.

The circuit cost is quite low, and the driver (Listing 1) reveals a lot about assembly language programming of the 6526 CIA. For CoCo, drive the control line (BIT0) with SERIAL OUT and connect BIT7 to CoCo's DC IN line. Program CD as an interrupt and eliminate the DELAY block in the flow chart shown in Figure 4. Figure 4 is the conversion flow chart for the converter of Figure 3. Adjust the sense of the control signals as necessary for your computer; BIT0 must be low for the ramp to run and BIT7 switches high when ramp coincidence occurs.

The premise of this experiment is that two I/O lines can interface with a simple A/D converter if the data conversion is internal to the computer. This makes the circuit work with an unmodified CoCo, and multiple A/Ds can be driven by the C64 User port. Data output on the C64 uses a small "window" in the lower left part of the screen. Binary data is converted to

ASCII (0-9) or to C64 screen characters (A-F). Writing the converted characters to $0770 thru $0773 puts data in the screen buffer, while $DB70 thru $DB73 are the corresponding Color RAM locations which make the data visible on the screen. Note also that writing $07 to the 6526 Control Register (lines 37-38 in Listing 1) sets Timer A or output on PB6; $41 sets Timer B for input on CNT2 (lines 28-29). Thus, PB6 and CNT2 must be connected by a jumper.

Once the circuit is built, vary the values of R2 and R3 to make the value shown on the screen vary between $FFFF for zero input and $FF80 for 5 volts. This accomplishes two thing: the circuit accuracy does not need more than 7 bits of resolution, and the number representing the voltage is restricted to a single byte for easier conversion to "real" numbers. Since this converter has a slightly non-linear output, the "classical" software correction would be to calibrate the circuit at (perhaps) ten points, and create a translation table. This technique uses a list of data (numbers read from the screen) and the corresponding voltage. So long as the non-linear output of a converter (or a sensor) is a smooth curve, a lookup table can noticeably increase accuracy. One other note: in common with many converters, this converter cannot convert negative input voltages.

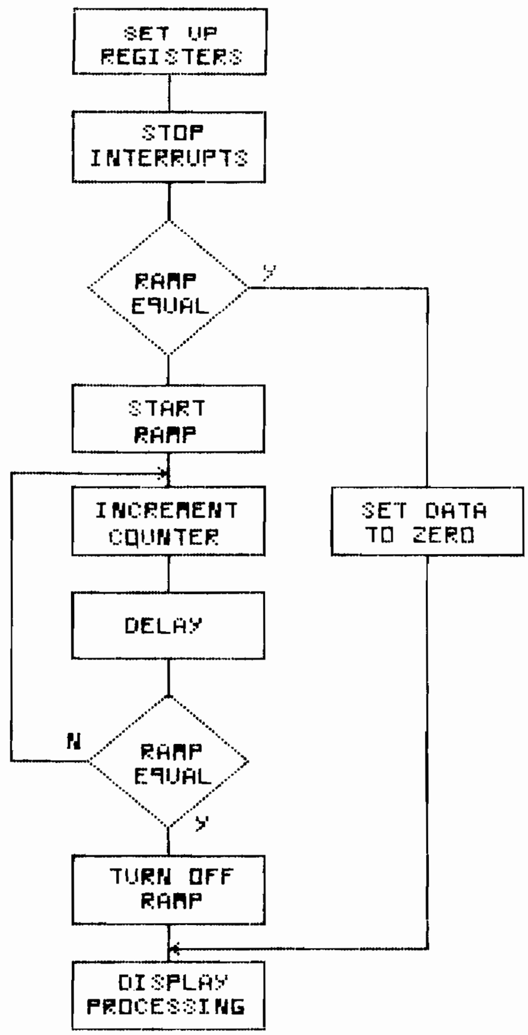


Figure 4. Flow chart for controlling A/D converter in Fig. 3.

Ralph Tenny may be corresponded with at P.O. Box 545, Richardson, TX 75080

MICRO

Commodore Compass

by Loren Wright

New Computers and Peripherals

At the January Consumer Electronics Show in Las Vegas, Commodore Business Machines announced two new computers and a number of peripherals. The two computers are the 264 and V364. Both include 64K of RAM (60K accessible for BASIC), the 7501 processor, built-in software capabilities, and four separate cursor keys. The specifications are similar to those of the Commodore 64, but it is doubtful that there will be much compatibility. There are 16 colors, each available in 8 luminances (similar to Atari). Only two music voices, no sprites, and only three graphics modes are available. The new BASIC 3.5 will include commands for graphics and sound, and there is a screen window capability. Also included is a built-in machine-language monitor.

Built-in software means that the programs are actually in ROM inside the computer, instantly available. Special models of the 264 will include The 264 Magic Desk, The 264 Word Processor, or an integrated package called The 264 3-PLUS-1.

The V364 includes voice synthesis capability, and extra commands in BASIC to support it. A 250-word vocabulary is built in, but you can add more words by loading them from disk or cassette. The V364 also includes a 19-key numeric keypad.

I am concerned that these machines use a 7501 processor. That means that Commodore will once again be introducing machines in the absence of compatible software, in spite of Commodore's assurance that a wide variety of software will be available at introduction. The lack of compatibility with the Commodore 64 is also unfortunate. It is quite possible that these machines, like the C128, may never make it to market. We shall see.

I doubt that these products will have any immediate impact on the C-64. It has been too successful for even Commodore to consider abandoning.

Peripherals announced at CES include a new disk drive (the 1542), a 60 cps dot-matrix printer (MPS 802), a color dot-matrix printer (MCS 801), and a daisy wheel printer (DPS 1101), all compatible with VIC, C64, and the two new computers. For the two new computers only are the SFS 481 fast disk drive, and the 1531 cassette drive.

Magic Voice is a voice synthesis cartridge for the Commodore 64 that includes a vocabulary of 235 phrases. More phrases can be loaded from disk or cassette. The Gorf and Wizard of Wor cartridges will be offered as talking cartridges, with more to come later. This unit adds capabilities similar to those included with the V364, but control will have to be with less-than-convenient BASIC V2 commands and machine language.

Jack Tramiel Resigns

Jack Tramiel, the founder and driving force of Commodore, has resigned, apparently in an attempt to make Commodore's management more efficient and structured. In his statement he cited personal reasons, but there is speculation that he was forced out. According to some sources this is the best thing that could happen to Commodore, but according to others it is the worst. I tend to think a little of both.

Jack has shown an incredible ability to think on his feet, making sudden, sweeping and unpredictable changes. His aggressive pricing policies have eliminated Texas Instruments from the home-computer market, and seriously hurt Atari, Apple, Timex-Sinclair, and others, putting Commodore at the top of the low-end microcomputer market. Middle and especially upper management has undergone so many changes that I've stopped keeping track.

Without Tramiel, Commodore will surely exhibit more stability, and probably more conservatism. Perhaps a higher priority will be assigned to things like customer and dealer support. However, Commodore may miss Tramiel's sixth-sense ability to react quickly and effectively to changes in the microcomputer market.

Listing 1

```
19000 REM READ IN ML PROGRAM
19010 MEM = 49152 : REM ADDRESS $C000
19020 READ XX
19030 IF XX < 256 THEN POKE MEM,XX:MEM = MEM+1:GOTO 19020
19040 RETURN
19500 DATA 0,38,1,32,52,192,32,80,192,32,65,192,32,80,192,173
19510 DATA 2,192,240,9,172,1,192,32,52,192,32,147,192,96,32,52
19520 DATA 192,32,104,192,32,65,192,32,104,192,172,0,192,173,2,192
19530 DATA 240,235,208,227,173,24,208,41,240,74,74,133,252,133,254,208
19540 DATA 6,169,216,133,252,133,254,169,0,133,251,169,1,133,253,96
19550 DATA 172,0,192,136,177,253,145,251,200,204,1,192,144,246,32,127
19560 DATA 192,165,251,201,232,208,233,96,172,1,192,177,251,145,253,136
19570 DATA 204,0,192,16,246,32,127,192,165,251,201,232,208,234,96,165
19580 DATA 251,24,105,40,133,251,133,253,230,253,165,252,105,0,133,252
19590 DATA 133,254,96,152,133,253,160,24,162,0,169,32,129,253,136,48
19600 DATA 15,24,165,253,105,40,133,253,165,254,105,0,133,254,208,234
19610 DATA 96,256
```

Sideways Screen Moves

Listing 1 is a basic-loader version of a lateral screen-move routine. A SYS 49155 instruction moves the entire screen left one character and a SYS 49182 moves it right.

What good is such a routine? One is in a screen editor, such as the one published in the November, 1983 issue of MICRO (66:28). Let's say you have carefully prepared a design on the screen, and you want to center it. It's easy if you can move the entire screen over.

The Commodore 64 offers a smooth-scrolling feature, whereby the entire screen can be moved in any direction in single-pixel increments. To make this look good, a number of things have to happen. The screen can be shrunk from 40 columns to 38, and from 25 rows to 24. This provides a hidden area where the new characters can be assembled before being scrolled on. However, when the screen reaches the 8-pixel limit of its fine scrolling capability, your programming must take over. The entire screen has to be shifted one character in the direction of the scroll--another use for my routine!

The routine is parameter driven. Three bytes at the beginning of the program control it: LCOL (49152) is the left-hand column to be moved; RCOL (49153) is the right-hand column to be moved; and FLAG (49154) determines whether to fill the vacated column with spaces (non-zero value) or to leave it as is. LCOL and RCOL must be in the range 0 to 39 and RCOL must be greater than LCOL. In addition, on a left move LCOL must be 1 or greater, and on a right move RCOL must 38 or less. The values in RCOL and LCOL will stay the same, so you can repeat calls without resetting them each time.

A little more on fine scrolling. It doesn't work quite the way it should. Switching from one end of the fine scrolling range to the other is so slow that it results in a noticeable screen jump. John Heilborn (Commodore 64 Graphics, Compute Books, 1983) resorts to using duplicate areas of screen memory. Everything is written on two screens and the two are switched back and forth. I certainly hope there is a better way. My routine will work with the smooth scrolling feature, but without some further refinements it will be far from smooth.

Listing 2

```
                ; SIDE SCROLL
                ; LOREN WRIGHT
                ; 20 FEB 1984
                ;
0400            SCRMEM  EQU $400
D018            VICMCR  EQU $D018
D800            CLRMEM  EQU $D800
                ;
00FB            PTRA    EQU $FB
00FD            PTRB    EQU $FD
                ;
C000                    ORG $C000
                ;
C000 00         LCOL    BYT 0
C001 26         RCOL    BYT 38
C002 01         FLAG    BYT 1
                ;
C003 20 34 C0   LINIT   JSR SCRSET
C006 20 50 C0           JSR MVLEFT
C009 20 41 C0           JSR CLRSET
C00C 20 50 C0           JSR MVLEFT
C00F AD 02 C0           LDA FLAG
C012 F0 09              BEQ QUIT
C014 AC 01 C0           LDY RCOL
C017 20 34 C0   SPCJMP  JSR SCRSET
C01A 20 93 C0           JSR SPCIN
C01D 60         QUIT    RTS
                ;
C01E 20 34 C0   RINIT   JSR SCRSET
C021 20 68 C0           JSR MVRGHT
C024 20 41 C0           JSR CLRSET
C027 20 68 C0           JSR MVRGHT
C02A AC 00 C0           LDY LCOL
C02D AD 02 C0           LDA FLAG
C030 F0 EB              BEQ QUIT
C032 D0 E3              BNE SPCJMP
                ;
C034 AD 18 D0   SCRSET  LDA VICMCR
C037 29 F0              AND
                        #%11110000
C039 4A                 LSR
C03A 4A                 LSR
C03B 85 FC              STA PTRA+1
C03D 85 FE              STA PTRB+1
C03F D0 06              BNE LOWSET
                ;
C041 A9 D8      CLRSET  LDA /CLRMEM
C043 85 FC              STA PTRA+1
C045 85 FE              STA PTRB+1
                ;
C047 A9 00      LOWSET  LDA #0
C049 85 FB              STA PTRA
C04B A9 01              LDA #1
C04D 85 FD              STA PTRB
C04F 60                 RTS
                ;
C050 AC 00 C0   MVLEFT  LDY LCOL
C053 88                 DEY
C054 B1 FD      LLOOP   LDA (PTRB),Y
C056 91 FB              STA (PTRA),Y
C058 C8                 INY
C059 CC 01 C0           CPY RCOL
C05C 90 F6              BCC LLOOP
C05E 20 7F C0           JSR BUMPRW
C061 A5 FB              LDA PTRA
C063 C9 E8              CMP #$E8
C065 D0 E9              BNE MVLEFT
C067 60                 RTS
                ;
C068 AC 01 C0   MVRGHT  LDY RCOL
C06B B1 FB      RLOOP   LDA (PTRA),Y
C06D 91 FD              STA (PTRB),Y
C06F 88                 DEY
C070 CC 00 C0           CPY LCOL
C073 10 F6              BPL RLOOP
C075 20 7F C0           JSR BUMPRW
C078 A5 FB              LDA PTRA
C07A C9 E8              CMP #$E8
C07C D0 EA              BNE MVRGHT
C07E 60                 RTS
                ;
C07F A5 FB      BUMPRW  LDA PTRA
C081 18                 CLC
C082 69 28              ADC #40
C084 85 FB              STA PTRA
C086 85 FD              STA PTRB
C088 E6 FD              INC PTRB
C08A A5 FC              LDA PTRA+1
C08C 69 00              ADC #0
C08E 85 FC              STA PTRA+1
C090 85 FE              STA PTRB+1
C092 60                 RTS
                ;
C093 98         SPCIN   TYA
C094 85 FD              STA PTRB
C096 A0 18              LDY #24
C098 A2 00              LDX #0
C09A A9 20      SLOOP   LDA #$20
C09C 81 FD              STA (PTRB,X)
C09E 88                 DEY
C09F 30 0F              BMI DONE
C0A1 18                 CLC
C0A2 A5 FD              LDA PTRB
C0A4 69 28              ADC #40
C0A6 85 FD              STA PTRB
C0A8 A5 FE              LDA PTRB+1
C0AA 69 00              ADC #0
C0AC 85 FE              STA PTRB+1
C0AE D0 EA              BNE SLOOP
C0B0 60         DONE    RTS
                ;
C0B1                    END
```

MICRO

6 x 6 = 36

4 x 4 = 16

9 x 9 = 81

On Multiplication The 6809 Versus the 6502

3 x 3 = 9

2 x 2 = 4

8 x 8 = 64

by Cornelis Bongers

Although it took some time, there is now finally a 6809 board (the REHAFLEX board) that works on both the Apple II and the Basis 108. This board is downwards compatible with the well known Mill Board (the main difference being an extensive memory mapping option, so that it supports the Apple Flex as well as the Apple OS-9 operating system).

When the board arrived, I enthusiastically started to learn 6809 machine language and I readily encountered the MUL instruction. With this instruction, two bytes stored in the A- and B-accumulator can be multiplied and the result is stored in the 16 bit D-accumulator. The latter is not a separate accumulator (low byte),but consists of the A-accumulator (high byte) and the B-accumulator (low byte), respectively. Since the 6502 lacks a MUL instruction, it seems an interesting experiment to substitute a 6809 floating point (FP) multiplication routine for the Applesoft 6502 FP multiplication routine, in order to speed things up a bit. I was especially motivated to undertake this experiment after I noticed how quickly FP multiplications are done in BASIC09. This article describes the result of the experiment and present a general and a special purpose multiplication routine for the 6809.

Figure 1

```
[BONGER-1.LST]
                ; DISASSEMBLY OF APPLESOFT'S
                ; FP MULTIPLICATION ROUTINE
                ;
009E            MFPM    EQU $9E     ; MANTISSA MAIN FP ACC
00A6            SFPM    EQU $A6     ; MANTISSA SEC FP ACC
0062            PROD    EQU $62     ; MANTISSA OF PRODUCT
00AC            EXTB    EQU $AC     ; EXTENSION BYTE
E8DA            SHIFT   EQU $E8DA   ; FAST SHIFT ROUTINE
EA0E            EXPO    EQU $EA0E   ; DETERMINE EXPONENT/SIGN
EAE6            NORM    EQU $EAE6   ; NORMALIZE ROUTINE
                ;
E982                    ORG $E982
                ;
E982 D0 03              BNE DOIT    ; BRANCH IF NON-ZERO EXPONENT
E984 4C E2 E9           JMP RETURN  ; ELSE, RETURN TO CALLER
                ;
                ; NOTE THAT THE STATEMENTS ABOVE
                ; CAN BE REPLACED BY: BEQ RETURN
                ;
```

FP Multiplication with the 6502

The Applesoft FP multiplication routine starts at $E982. Prior to invocation, the main floating point accumulator (MFP, at $9D-$A1 and the extension byte at $AC) and the secondary floating point accumulator (SFP, at $A5-$A9) must have been loaded with the numbers that have to be multiplied. Furthermore, the value of the MFP exponent must be loaded in the 6502 accumulator just before the subroutine call to $E982 is issued. (For more details see: In the Heart of Applesoft, MICRO No. 33, February 1981). The routine (see Figure 1) starts with a BNE instruction. This has the effect that the multiplication is only carried out if the MFP differs from zero. A zero MFP exponent indicates that the whole number is zero and, in that case, control returns to caller immediately.

The (preliminary) value of the exponent of the product and the sign of the mantissa are determined by the subroutine at $EAOE, which is called at $E987. Next, the multiplication of the mantissas is done. It is this part of the multiplication routine that is best suited for 6809 code substitution. As can be seen from Figure 1, locations $62-$65 are initialized to zero first. These locations will further be referred to as the 'product accumulator'. The product accumulator is used to build up the product and when the mantissa multiplication is completed, it is transferred to the MFP. The MFP is then normalized and control returns to caller.

Broadly speaking, the mantissa multiplication is performed by examining the bits of the MFP mantissa (further to be called the MFPM) one by one, beginning with the last bit of the extension byte ($AC). If a bit is set,the SFPM is added to the product accumulator and next, the product accumulator is shifted one bit to the right (i.e., divided by two). If the bits is not set, only the shifting to the right takes place (thus no adding). The actual code of the routine consists of a main driver and a subroutine. The main driver loads the bytes of the MFPM, one by one, in the 6502 accumulator (beginning with the last byte) and calls the subroutine at $E9BO. This subroutine handles the bit examination and builds up the product accumulator. Note that if the subroutine is entered with the zero flag set, indicating that the byte is zero, control is transferred to location $E8DA. At this address resides code that shifts the product accumulator one byte to the right (i.e., a division by 256). This obviously goes faster than shifting the product accumulator eight times on bit level. If the byte is not zero, the process above (adding and shifting) takes place. Note the clever method that is used to create a loop that is executed 8 times (see the instructions at $E9B5-$E9B6 and $E9DE-$E9EO).

```
E987 20 0E EA   DOIT     JSR EXPO     ; DETERMINE NEW EXPO/SIGN
E98A A9 00               LDA #$00     ; INIT PRODUCT ACC TO 0
E98C 85 62               STA PROD
E98E 85 63               STA PROD+1
E990 85 64               STA PROD+2
E992 85 65               STA PROD+3
E994 A5 AC               LDA EXTB     ; START WITH EXTENSION BYTE
E996 20 B0 E9            JSR MANMUL   ; AND MULTIPLY WITH SFPM
E999 A5 A1               LDA MFPM+3
E99B 20 B0 E9            JSR MANMUL
E99E A5 A0               LDA MFPM+2   ; DO OTHER BYTES NEXT
E9A0 20 B0 E9            JSR MANMUL
E9A3 A5 9F               LDA MFPM+1
E9A5 20 B0 E9            JSR MANMUL
E9A8 A5 9E               LDA MFPM
E9AA 20 B5 E9            JSR MANMUL1
E9AD 4C E6 EA            JMP NORM     ; NORMALIZE AND EXIT
                ;
E9B0 D0 03      MANMUL   BNE MANMUL1  ; BRANCH IF BYTE<> 0
E9B2 4C DA E8            JMP SHIFT    ; ELSE, SHIFT FAST
E9B5 4A         MANMUL1  LSR          ; GET LAST BIT IN CARRY
E9B6 09 80               ORA #$80     ; SET FIRST BIT
E9B8 A8         LOOP     TAY          ; SAVE IN Y-REG
E9B9 90 19               BCC NOADD    ; OMIT ADD IF BIT NOT SET
E9BB 18                  CLC
E9BC A5 65               LDA PROD+3   ; ADD SFPM TO PRODUCT ACC
E9BE 65 A9               ADC SFPM+3
E9C0 85 65               STA PROD+3
E9C2 A5 64               LDA PROD+2
E9C4 65 A8               ADC SFPM+2
E9C6 85 64               STA PROD+2
E9C8 A5 63               LDA PROD+1
E9CA 65 A7               ADC SFPM+1
E9CC 85 63               STA PROD+1
E9CE A5 62               LDA PROD
E9D0 65 A6               ADC SFPM
E9D2 85 62               STA PROD
E9D4 66 62      NOADD    ROR PROD     ; DIVIDE PRODUCT ACC BY 2
E9D6 66 63               ROR PROD+1
E9D8 66 64               ROR PROD+2
E9DA 66 65               ROR PROD+3
E9DC 66 AC               ROR EXTB     ; EXTB = LAST BYTE PROD ACC
E9DE 98                  TYA          ; GET (SHIFTED) BYTE MFPM
E9DF 4A                  LSR          ; SHIFT ONCE AGAIN
E9E0 D0 D6               BNE LOOP     ; LOOP 8 TIMES
E9E2 60         RETURN   RTS
```

The time critical part of the code ($E9BC-$E9DC) is written out completely in order to minimize execution time. The execution time varies, of course, since it depends on the number of non-zero bytes in the MFP and the number of bits set in the MFPM. Multiplying two small integers takes generally much less time than multiplying two fractional numbers, because the mantissa of an (Applsoft) integer number contains at most two non-zero bytes. When we assume that on the average half of the bits are set in the MFPM (including the extension byte), the number of cycles needed for the mantissa multiplication will about 2250. Since a 6502 cycle corresponds approximately to a microsecond, we arrive at 2.3 milliseconds per mantissa multiplication.

Multiplication with the 6809

The availability of the MUL instruction opens the way to use an entirely different multiplication routine. Rather than working on bit level, we can now do things on byte level. A possible approach is outlined in Figure 2. First, byte 4 of the MFPM (byte i of the MFPM will further be referred to as MFPMi) is multiplied with SFPM3 and the result is put in bytes 7 and 8 of the (zero-initialized) product accumulator. Next, MFPM3 is multiplied with SFPM3 and the result is added to bytes 6 and 7 of the product accumulator. This process is continued until all bytes of the MFPM are multiplied with SFPM3. The following step is to multiply all bytes of the MFPM with SFPM2. We start with SFPM2 * MFPM4 and add the result to bytes 6 and 7 of the product accumulator. But now a problem arises. Namely, when the addition is executed, a carry may be generated, for the product accumulator already contains the results of MFPM * SFPM3. If a carry is generated, byte 5 of the product accumulator must be incremented by one. The latter operation may, however, also generate a carry (actually: set the zero flag), which would mean that byte 4 must also be incremented, and so on. This is certainly a drawback of this approach, for a considerable amount of time may be involved with the carry-processing. An additional drawback is that the product accumulator must consist of 9 bytes. This can be reduced to 6 bytes, but then the entire product accumulator must be shifted one byte to the right after each multiplication of the MFP mantissa with a byte of the SFPM.

Another approach that looks more promising works as follows. In algebraical terms, the 20 (4x5) byte by byte multiplications required to multiply the two mantissas can be split up in a number of groups. Each group consists of those 'partial products' that have the same exponent. When the MFPM is looked at as a binary number, it can be represented as follows:

```
MFPM=MFPM0*2^0 + MFPM1*2^8 + MFPM2*2^16
                           + MFPM3*2^24 + MFPM4*2^32
```

Multiplication with the SFMP, i.e.,

```
SFPM=SFPM0*2^0 + SFPM1*2^8 + SFPM2*2^16 + SFPM3*2^24
```

gives:

```
(SFPM0*MFPM0)*2^0 +
(SFPM0*MFPM1+SFPM1*MFPM0)*2^8 +
(SFPM0*MFPM2+SFPM1*MFPM1+SFPM2*MFPM0)*2^16 +
(SFPM0*MFPM3+SFPM1*MFPM2+SFPM2*MFPM1+SFPM3*MFPM0)*2^24 +
(SFPM0*MFPM4+SFPM1*MFPM3+SFPM2*MFPM2+SFPM3*MFPM1)*2^32 +
(SFPM1*MFPM4+SFPM2*MFPM3+SFPM3*MFPM2)*2^40 +
(SFPM2*MFPM4+SFPM3*MFPM3)*2^48 +
(SFPM3*MFPM4)*2^56
```

The product can be built up in 8

Figure 2. Multiplication with the 6809 (method 1)

```
-----------------------------------------
: MFPM0 : MFPM1 : MFPM2 : MFPM3 : MFPM4 :  MFPM
-----------------------------------------
        ---------------------------------
        : SFPM0 : SFPM1 : SFPM2 : SFPM3 :  SFPM
        ---------------------------------
----------------------------------------- x
                        -----------------
                        : SFPM3 * MFPM4 :
                        -----------------
                -----------------
                : SFPM3 * MFPM3 :
                -----------------
                ------------------------- +
                -------------------------
                : PROD6 : PROD7 : PROD8 :
                -------------------------
        -----------------
        : SFPM3 : MFPM2 :
        -----------------
        --------------------------------- +
        ---------------------------------
        : PROD5 : PROD6 : PROD7 : PROD8 :
        ---------------------------------
          etc.
```

Figure 3. Multiplication with the 6809 (method 2)

```
-----------------------------------------
: MFPM0 : MFPM1 : MFPM2 : MFPM3 : MFPM4 :  MFPM
-----------------------------------------
        ---------------------------------
        : SFPM0 : SFPM1 : SFPM2 : SFPM3 :  SFPM
        ---------------------------------
----------------------------------------- x
                        -----------------
                        : SFPM3 * MFPM4 :
                        -----------------
                             \
                                 ---------
                                 : PROD4 :
                                 ---------
                        -----------------
                        : SFPM2 * MFPM4 :
                        -----------------
                        ----------------- +
                        -----------------
                        : PROD3 : PROD4 :
                        -----------------
                        -----------------
                        : SFPM3 * MFPM3 :
                        -----------------
                ------------------------- +
                : PROD2 : PROD3 : PROD4 :
                -------------------------
                          \       \
                        -----------------
                        : PROD3 : PROD4 :
                        -----------------
                        -----------------
                        : SFPM1 * MFPM4 :
                        -----------------
                ------------------------- +
                -------------------------
                : PROD2 : PROD3 : PROD4 :
                -------------------------
                      etc.
```

Figure 4

```
                * MULTIPLY BINARY NUMBERS
                *  ON A 6809
                *
                * BY CORNELIS BONGERS
                * APRIL 1983, VERSION 1.1
                *  IN MICRO #70, MARCH 1984
                *
                * NOTES
                *
                * LPROD MUST BE >=3 AND <= LMPL+LMPC
                * LMPC MUST BE >= LMPL
                * LMPL AND LMPC MUST EACH BE < 128
                * LMPL+LMPC-LPROD MUST BE <128
                *
                * INITIALIZATION
                *
           00A6 MPL     EQU   $A6      START MULTIPLIER
           009E MPC     EQU   $9E      START MULTIPLICANT
           0062 PROD    EQU   $62      START PRODUCT
           0004 LMPL    EQU   $4       LENGTH MULTIPLIER
           0005 LMPC    EQU   $5       LENGTH MULTIPLICANT
           0005 LPROD   EQU   $5       LENGTH PRODUCT
                *
                * TEMPORARY REGISTERS
                *
           0006 CURX    EQU   $6       POINTER TO MULTIPLIER
           0008 CURY    EQU   $8       POINTER TO MULTIPLICANT
           00FB ITCNT   EQU   $FB      NO. OF ITERATIONS
           00FC ITER    EQU   $FC      ITERATION COUNTER
           00FD SGNCNT  EQU   $FD      NO. OF SHIFTS TO THE RIGHT
           00FE SAME    EQU   $FE      NO. OF LONGEST ITERATIONS
           00FF MAILBOX EQU   $FF      USED FOR 6502 COMMUNICATION
                *
                * START OF PROGRAM
                *
0000 0F   FF    START   CLR   MAILBOX
0002 96   FF    WAIT    LDA   MAILBOX  WAIT FOR MULTIPLY COMMAND
0004 2A   FC            BPL   WAIT
0006 8E   00A9          LDX   #MPL+LMPL-1 POINTS TO END OF MULTIPLIER
0009 108E 00A3          LDY   #MPC+LMPC POINTS TO END+1 OF MULTIPLICANT
000D 109F 08            STY   CURY
0010 86   04            LDA   #LMPL+LMPC-LPROD SET UP SAME AND
0012 C6   01            LDB   #LMPC-LMPL SGNCNT
0014 DD   FD            STD   SGNCNT
0016 CE   0065          LDU   #PROD+LPROD-2 POINTS TO END-1 OF PRODUCT
0019 6F   C4            CLR   ,U       CLEAR LAST BYTES OF PROD. ACCUM.
001B 6F   41            CLR   1,U
001D 0F   FB            CLR   ITCNT    SET NO. OF ITERATIONS OT 0
001F 20   1B            BRA   ENTRY1   GO MULTIPLY
0021 33   5F    ALIGN   LEAU  -1,U     START MAIN LOOP
0023 9E   06    CONT    LDX   CURX
0025 8C   00A7          CPX   #MPL+1   START MULTIPLIER REACHED ?
0028 24   0E            BHS   NXTPS0   BRANCH IF NOT
002A 0A   09            DEC   CURY+1   UPDATE PTR TO MULTIPLICANT
002C 0A   FE            DEC   SAME     KEEP TRACK OF MAX # ITERATIONS
002E 2A   0E            BPL   NXTPS1
0030 0A   FB            DEC   ITCNT    DOWN THE HILL
0032 27   0C            BEQ   SKPCLR   LAST PASS
0034 2A   08            BPL   NXTPS1
0036 20   C8            BRA   START    READY
```

iterations, where each iteration corresponds to the calculation of one of the terms between parenthesis above. We start with the calculation of the last term (i.e., SFPM3*MFPM4). This involves a single multiplication and the result is put in bytes 7 and 8 of the product accumulator. Next, the last but one term is computed. This involves two multiplications, and both results are added to bytes 6 and 7 of the product accumulator. A carry may result here, but since bytes 0-5 of the product accumulator are (still) zero, a simple increment of byte 5 will do. This increment can never generate a carry, for, as can be figured out, a (second) carry can only occur if a term involves more than 257 multiplications. The other terms are calculated and processed in a similar way.

The advantages of the approach above are that no extended carry-processing is necessary and that we need not reserve the full length of the product accumulator (i.e., 9 bytes). If we want to reserve only 5 bytes for the mantissa of the product accumulator (as is the case in Applesoft), we set the product accumulator pointer, which references the current byte(s) of the product accumulator, to byte 3 during the first 5 iterations. Consequently, after the calculation of each of the first four terms the two (!) relevant bytes of the product accumulator must be shifted one byte to the right (see Figure 3 for an illustration). Only after the calculation of the 5th term (and each of the remaining terms), the product accumulator pointer is decremented and the shift to the right operation is omitted.

The routine listed in Figure 4 shows the code for the multiplication process discussed above. Although no stack-wise parameter passing is employed, the routine is set up in such a way that it can easily be adapted to non-Applesoft applications. The length (in bytes) of the multiplicant, the multiplier and the product can be specified by the user (see initialization section), provided the restrictions mentioned in the listing are satisfied.

The 6502 Versus the 6809

The final step is linking the 6809 multiplication routine to Applesoft. The 6502 driver, which takes care of 6809 - 6502 communications is displayed in Figure 5. Since the 6809 routine expects a contiguous MFP accumulator, the value of the extension byte ($AC) is moved to $A2. The old value of $A2 is temporarily stored in the X-register. A similar save/restore operation is performed on location $66, which corresponds to the 'extension' byte of the product accumulator.

Installation of the 6809 routine involves the following steps:

1) Load the 6809 code at a suitable address in memory, for example at $9400
2) Coldstart the 6809 with the 6809 reset vector set to the address above. Next put the 6809 in HALT state.
3) Move Applesoft into the Language Card and read/write enable the Language Card
4) Load the 6502 driver at $E98A
5) Coldstart Applesoft and set HIMEM to the address specified at step 1

After performing step 5, all multiplications will be done by the 6809. A good method to check if things work all right is to write a small Applesoft program that compares the results of two multiplications, the first computed under 'normal' Applesoft and the second computed under the 6809 version (by means of PEEK(49280) and PEEK(49281), the Language Card can be switched on and off from BASIC).

For the speed comparisons I used the program displayed in Figure 6. The program took 52 secs to compute the 10000 multiplications with normal Applesoft. Next I switched to the 6809 version and ran the program again, but to my great disappointment, the reduction in execution time was only 4 secs. First, I thought there had to be an error somewhere (in the form of a temporary hang-up of either the program or my watch), since the same program ran in 21 secs under BASIC09. However, I was unable to find any bugs, so I decided to establish more precise timing results for the multiplication operation. This can be done rather easily by moving a fresh copy of Applesoft into the Language Card and inserting a JMP $EAE6 instruction (to the normalize routine) at $E98A, thereby eliminating the mantissa multiplication. This led to an execution time (of the program in Figure 6) of 32 secs. Deducting this from the 52 secs realized earlier, we arrive at a time of 2 millisecs per multiplication. With the 6809, the time needed for a multiplication is then

```
0038 30   1F       NXTPS0  LEAX  -1,X     UPDATE PTR TO MULTIPLIER
003A 0C   FB               INC   ITCNT    UP THE HILL
003C 9F   06       ENTRY1  STX   CURX     UPDATE CURX
003E 6F   5F       NXTPS1  CLR   -1,U     CLEAR OVERFLOW BYTE
0040 109E 08       SKPCLR  LDY   CURY     GET PTR TO MULTIPLICANT
0043 96   FB               LDA   ITCNT    GET # OF ITERATIONS FOR THIS PASS

0045 97   FC               STA   ITER     SET UP ITERATION COUNTER
0047 A6   80       GOMUL   LDA   ,X+      GET BYTE OF MULTIPLIER
0049 E6   A2               LDB   ,-Y      AND MULTIPLICANT
004B 3D                    MUL            MULTIPLY
004C E3   C4               ADDD  ,U       ADD TO PARTIAL PRODUCT
004E ED   C4               STD   ,U       AND STORE
0050 24   02               BCC   NOOVER
0052 6C   5F               INC   -1,U
0054 0A   FC       NOOVER  DEC   ITER
0056 2A   EF               BPL   GOMUL
0058 D6   FD               LDB   SGNCNT   UPDATE U-PTR ?
005A 27   C5               BEQ   ALIGN    BRANCH IF SO
005C 0A   FD               DEC   SGNCNT
005E EC   5F               LDD   -1,U     SHIFT PARTIAL PRODUCT
0060 ED   C4               STD   ,U
0062 20   BF               BRA   CONT     CONTINUE
                   *
                   * END OF PROGRAM
```

Figure 5

```
                ; 6502 DRIVER FOR 6809 FP MULTIPLICATION
                ;
009E             MFPM     EQU $9E     ; MANTISSA MAIN FP ACC
0062             PROD     EQU $62     ; MANTISSA OF PRODUCT
00AC             EXTB     EQU $AC     ; EXTENSION BYTE
00FF             MAILBOX  EQU $FF     ; FOR 6502/6809 COMM
EAE6             NORM     EQU $EAE6   ; NORMALIZE ROUTINE
C0B1             HLT6809  EQU $C0B1   ; HALT/START 6809
                ;
E98A                      ORG $E98A
                ;
E98A A6 A2                LDX MFPM+4  ; SAVE $A2
E98C A4 66                LDY PROD+4  ; AND LAST BYTE PRODUCT
E98E A5 AC                LDA EXTB    ; PROVIDE 6809 WITH
E990 85 A2                STA MFPM+4  ; A CONTINUOUS MFPM
E992 A9 80                LDA #$80    ; PREPARE CALL
E994 85 FF                STA MAILBOX
E996 8D B1 C0             STA HLT6809 ; START 6809
E999 A5 FF      WAIT      LDA MAILBOX
E99B 30 FC                BMI WAIT    ; WAIT UNTIL 6809 IS READ
E99D 8D B1 C0             STA HLT6809 ; HALT 6809
E9A0 A5 66                LDA PROD+4  ; SET EXTENSION BYTE
E9A2 85 AC                STA EXTB    ; PROPER VALUE
E9A4 86 A2                STX MFPM+4  ; RESTORE CLOBBERED
E9A6 84 66                STY PROD+4  ; LOCATIONS
E9A8 4C E6 EA             JMP NORM    ; NORMALIZE AND EXIT
                ;
E9AB                      END
```

Figure 6

```
5 REM (SIMPLE) PROGRAM FOR SPEED COMPARISONS
10 LET A = 5 / 3: B = 5 / 3
20 FOR I = 1 TO 10000: C = A * B: NEXT
30 PRINT CHR$(7): REM BELL
```

Figure 7

```
                  * SPECIAL PURPOSE MULTILICATION ROUTINE
                  * TO SPEED UP APPLESOFT

                  * MFP MANTISSA STARTS AT $93 (5 BYTES)
                  * SRP MANTIXXA STARTS AT $A6 (4 BYTES)
                  * PRODUCT MANTIXXA STARTS AT $64 (5 BYTES)
                  *
                  MAILBOX EQU $FF USED FOR 6502 COMMUNICATION
                  *
                  MULTDR MACRO CONTROLS GENERATION OF MULT SEGMENTS
                  PNT1   SET   &1      SETS PTRS TO MANTISSA BYTES
                  PNT2   SET   &2
                  PNTPR  SET   $62+&1+&2-$9E-$A6-&3  SET PRODUCT PTR
                  CARRY  SET   &4      CARRY PROCESSING FLAG
                  MULT   PNT1,PNT2,PNTPR,CARRY  INSERT MULT SEGMENT
                  CARRY  SET   0       TURN CARRY PROCESSING ON
                  PNT1   SET   PNT1+1
                         IFN   (PNT1-$AA),2  LOOP UNTIL DONE
                  PNT2   SET   PNT2-1
                         IFN   (PNT2-$9D),-5
                         ENDM
                  *
                  MULT   MACRO GENERATES MULT SEGMENT
                         LDA   &2      GET BYTES TO BE MULTIPLIED
                         IFN   (&2=$A2)  CHECK IF EXTENSION BYTE AND
                         BEQ   *+13-&4   GENERATE BRANCH INSTR IF SO
                         LDB   &1
                         MUL
                         ADDD  &3      ADD TO PARTIAL RESULT
                         STD   &3
                         IF    &4=4,2  SUPPRESS CARRY CHECK
                         BCC   *+4     IF &4<>0
                         INC   &3-1
                         ENDM
                  *
                  * MAIN PROGRAM
                  *
0000 0F   FF      START  CLR   MAILBOX
0002 96   FF      WAIT   LDA   MAILBOX WAIT FOR MULT COMMAND
0004 2A   FC             BPL   WAIT
0006 8E   0000           LDX   #$0
0009 9F   62             STX   $62     INIT BYTE 0 AND 1 OF PRODUCT
000B 96   A2             LDA   $A2     ACCUM. CHECK EXTN. BYTE
000D 26   34             BNE   ALL     GO ALL THE WAY IF <>0
000F 97   66             STA   $66     INIT LAST BYTE PRODUCT ACCUM
0011 109E A0             LDY   $A0     DEALING WITH INTEGERS ?
0014 26   32             BNE   NOINT   BRANCH IF NOT
0016 109E A8             LDY   $A8     DITTO
0019 26   2D             BNE   NOINT
                  *
                  * INTEGER MULTIPLICATION
                  * (4 BYTE BY BYTE MULTIPLICATIONS)
                  *
001B 96   A7             LDA   $A7
001D D6   9F             LDB   $9F
001F 3D                  MUL
0020 DD   64             STD   $64
0022                     MULT  $A6,$9F,$63,4
002B                     MULT  $A7,$9E,$63,0 DO CARRY CHECK HERE
0038                     MULT  $A6,$9E,$62,4
0041 20   BD             BRA   START
                  *
```

4.8-3.2=1.6 millisecs, so the increase in 'multiplication' performance is 20%. An aspect that also must be take into consideration concerns the extension byte. The extension byte is only us to prevent losing precision during the evaluation of expressions. However, in our test program the extension byte will always be zero, since no temporary results are generated. The 6502 multiplication routine skips in this case 8 add/shift operations on bit level and shifts the product accumulator one byte to the right instead. A similar thing does not happen in the 6809 multiplication routine; here the 20 multiplications are executed always. Consequently, the 6502 has an inherent advantage, which results in a gain of 3-4 secs. Adding this to the 20 secs obtained above, we have 2.3-2.4 millisecs per 6502 FP mantissa multiplication. This agrees with the cycle-based time calculation in section 2. The determination of the number of cycles consumed by the 6809 multiplication routine is easy with the help of the excellent FLEX debugger; it came up with 1560 cycles. Adding the 6502 driver overhead (i.e., 46 cycles), the 6809 FP mantissa multiplication should, therefore, last about 1.6 millisecs.

So, overall the cycle times fit reasonably well with the timing results. When multiplying fractional numbers, you can expect a speed improvement of about 10%-20% per multiplication. In the case of integers, however, you will be faced with a significant slowdown in execution speed. For example, the program above with A and B set to 10, rather than to 5/3, needs only .9 millisecs per 6502 multiplication, but (still) 1.6 millisecs per 6809 multiplication.

The results above strongly suggest that implementation of a 6809 multiplication routine in Applesoft is not attractive. There is still hope though, for BASIC09 somehow manages to execute FP multiplication in much less time. So, the question is: How come the BASIC09 multiplication routine is so fast. After delving into the BASIC09 interpreter to locate the FP multiplication routine, the answer appeared to be easy. In the first place, BASIC09 doesn't use an extension byte. This means that an FP multiplication consists of 16 rather than 20 byte by byte multiplications. In the second place,it appeared that the entire multiplication routine is written

```
                   * FLOATING POINT MULTIPLICATION
                   * (16 OR 20 BYTE BY BYTE MULT.)
                   *
0043 D6   A9       ALL    LDB   $A9
0045 3D                   MUL
0046 97   66              STA   $66
0048 9F   64       NOINT  STX   $64     INIT BYTE 2 AND 3 OF PRODUCT ACC
004A                      MULTDR $A8,$A2,3,4  GENERATE 2 SEGMENTS
0062                      MULTDR $A7,$A2,3,0  GENERATE 3 SEGMENTS
008B                      MULTDR $A6,$A2,3,0  GENERATE 4 SEGMENTS
00C1 DC   62              LDD   $62     THROW EVERYTHING AWAY EXCEPT
00C3 DD   65              STD   $65     THE MOST SIGNIFICANT BYTES
00C5 9F   62              STX   $62     REINIT PRODUCT ACC.
00C7 0F   64              CLR   $64
00C9                      MULTDR $A6,$A1,0,0  GENERATE 4 SEGMENTS
00FD                      MULTDR $A6,$A0,0,0  GENERATE 3 SEGMENTS
0124                      MULTDR $A6,$9F,0,0  GENERATE 2 SEGMENTS
013E                      MULTDR $A6,$9E,0,4  GENERATE 1 SEGMENT
0147 16   FEB6            LBRA  START
```

Figure 8

```
                   * SAMPLE MACRO EXPANSION
0000                      ORG   $0000
                   *
                   *  DEFINE MULTDR MACRO
                   *
                   MULTDR MACRO
                   PNT1   SET   &1
                   PNT2   SET   &2
                   PNTPR  SET   $62+&1+&2-$9E-$A6-&3
                   CARRY  SET   &4
                          MULT  PNT1,PNT2,PNTPR,CARRY
                   CARRY  SET   0
                   PNT1   SET   PNT1+1
                          IFN   (PNT1-$AA),2
                   PNT2   SET   PNT2-1
                          IF    (PNT2-$9D),-5
                          ENDM
                   *
                   *  DEFINE MULT MACRO
                   *
                   MULT   MACRO
                          LDA   &2        GET BYTES TO BE MULTIPLIED
                          IFN   (&2=$A2)  CHECK IF EXTENSION BYTE AND
                          BEQ   *+13-&4   GENERATE BRANC INSTR IF SO
                          LDB   &1
                          MUL
                          ADDD  &3        ADD TO PARTIAL RESULT
                          STD   &3
                          IF    &4=4,2    SUPPRESS CARRY CHECK
                          BCC   *+4       IF &4<>0
                          INC   &3-1
                          ENDM
                   *
                   *  SAMPLE EXPANSION OF MULT MACRO
                   *
0000                      MULT  $A6,$9F,$63,4
0000 96   9F              LDA   $9F       GET BYTES TO BE MULTIPLIED
0002 27   07              BEQ   *+13-4    GENERATE BRANCH INSTR IF SO
0004 D6   A6              LDB   $A6
0006 3D                   MUL
0007 D3   63              ADDD  $63       ADD TO PARTIAL RESULT
0009 DD   63              STD   $63
                          ENDM
```

(Continued on next page)

Figure 8. Note: This is a sample of the expanded MACRO's: MULT and MULTDR.

```
                        * SAMPLE EXPANSION OF MULTDR MACRO
                        *
000B                            MULTDR $A6,$A1,0,0
              00A6  PNT1        SET    $A6
              00A1  PNT2        SET    $A1
              0065  PNTPR       SET    $62+$A6+$A1-$9E-$A6-0
              0000  CARRY       SET    0
000B                            MULT   PNT1,PNT2,PNTPR,CARRY
000B 96   A1                    LDA    PNT2      GET BYTES TO BE MULTIPLIED
000D 27   0B                    BEQ    *+13-CARRY GENERATE BRANC INSTR IF SO
000F D6   A6                    LDB    PNT1
0011 3D                         MUL
0012 D3   65                    ADDD   PNTPR     ADD TO PARTIAL RESULT
0014 DD   65                    STD    PNTPR
0016 24   02                    BCC    *+4       IF CARRY<>0
0018 0C   64                    INC    PNTPR-1
                                ENDM
              0000  CARRY       SET    0
              00A7  PNT1        SET    PNT1+1
              00A0  PNT2        SET    PNT2-1
001A                            MULT   PNT1,PNT2,PNTPR,CARRY
001A 96   A0                    LDA    PNT2      GET BYTES TO BE MULTIPLIED
001C 27   0B                    BEQ    *+13-CARRY GENERATE BRANC INSTR IF SO
001E D6   A7                    LDB    PNT1
0020 3D                         MUL
0021 D3   65                    ADDD   PNTPR     ADD TO PARTIAL RESULT
0023 DD   65                    STD    PNTPR
0025 24   02                    BCC    *+4       IF CARRY<>0
0027 0C   64                    INC    PNTPR-1
                                ENDM
              0000  CARRY       SET    0
              00A8  PNT1        SET    PNT1+1
              009F  PNT2        SET    PNT2-1
0029                            MULT   PNT1,PNT2,PNTPR,CARRY
0029 96   9F                    LDA    PNT2      GET BYTES TO BE MULTIPLIED
002B 27   0B                    BEQ    *+13-CARRY GENERATE BRANC INSTR IF SO
002D D6   A8                    LDB    PNT1
002F 3D                         MUL
0030 D3   65                    ADDD   PNTPR     ADD TO PARTIAL RESULT
0032 DD   65                    STD    PNTPR
0034 24   02                    BCC    *+4       IF CARRY<>0
0036 0C   64                    INC    PNTPR-1
                                ENDM
              0000  CARRY       SET    0
              00A9  PNT1        SET    PNT1+1
              009E  PNT2        SET    PNT2-1
0038                            MULT   PNT1,PNT2,PNTPR,CARRY
0038 96   9E                    LDA    PNT2      GET BYTES TO BE MULTIPLIED
003A 27   0B                    BEQ    *+13-CARRY GENERATE BRANC INSTR IF SO
003C D6   A9                    LDB    PNT1
003E 3D                         MUL
003F D3   65                    ADDD   PNTPR     ADD TO PARTIAL RESULT
0041 DD   65                    STD    PNTPR
0043 24   02                    BCC    *+4       IF CARRY<>0
0045 0C   64                    INC    PNTPR-1
                                ENDM
              0000  CARRY       SET    0
              00AA  PNT1        SET    PNT1+1
                                ENDM
                        *
                                END
```

out completely. This means that all loop and pointer overhead - which accounts for roughly 50% of the execution time - is eliminated. As a result, the BASIC09 mantissa multiplication requires, on the average, only about 750 cycles o.75 millisecs per mantissa multiplication.

The best way to improve FP multiplication seems, therefore, to use the BASIC09 approach. Though this takes many bytes of code, the increase in performance is impressive. Figure 7 displays the listing of a source file that cane used to generate a loop- and pointerless multiplication routine. The file can be assembled with the FLEX assembler and installed with the 5 steps outlined above. Two macro's have been defined; the first (MULTDR) controls the generation of the byte by byte multiplication segments and the second (MULT) generates the multiplication segments itself. As can be seen, the mantissas of MFP and SFP are checked on zero-bytes. If the last two bytes of both mantissas are zero, a fast integer multiplication routine is used.

The execution time of the routine is 39 secs for the program in Figure 6. That means 3.9-3.2=.7 millisecs per multiplication, implying a speed improvement of 65% relative to Applesoft. When setting A and B to 10, a multiplication takes only .2 millisecs, so integer multiplication is improved by more than 75%. The routine also speeds up other Applesoft functions. For example, the computation time for the SIN function is reduced by approximately 40% and the computation time for the SQR function by about 45%.

Conclusion

The best way to significantly speed up Applesoft multiplication with the 6809 is to use a fully expanded multiplication routine. Such a routine consumes a lot of memory, but the increase in performance (a 65%-75% reduction in the execution time of a multiplication) gives a good pay-off.

Cornelis Bongers may be reached at Erasmus University, Postbox 1738, 3000 DR Rotterdam, The Netherlands.

MICRO

Compile Your BASIC Subroutines

by Ann Marie Lancaster and Cliff Long

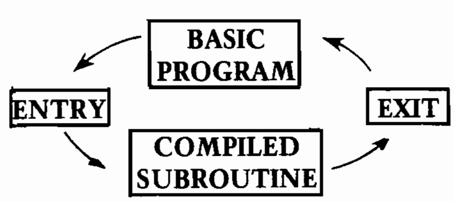
Interpreted BASIC is easy to use but slow, and Compiled BASIC is fast but difficult to use. This solution combines the best of both and works with machine language, too

Not only is BASIC a readily available language for micros, but its interpretive nature offers some advantages for program development. Unfortunately, execution speed is not one of these advantages. Compilers are now readily available for BASIC, and we have been using the Microsoft version with our Apple II's. The execution time for a specific surface plotting program (with hidden line removal) dropped from about three minutes to thirty-three seconds when compiled. Such time reductions seem typical of our applications and certainly lead to substantial savings for regularly used programs. The one time compilation does take a few minutes longer than a regular program execution.

If the total surface plotting program is compiled, then each new surface description requires a new compile step because function changes require a self-modification of the BASIC program [1]. Since, for classroom graphics applications, it is convenient to modify the defining function frequently depending on the homework problems or the whim of the questioning students, we decided to compile the slowest subroutine (the hidden line part) of the program. This allows us to change the function and viewing direction many times during a typical program execution without having to recompile. Hence, while part of the main program is running in interpreted BASIC, the slowest portion has been replaced by the fast running compiled form, thus giving us a very efficient "canned program" running at close to the machine code speed (without the tedious machine language programming). *[Ed. Note: Of course, if you enjoy machine language programming, the technique discussed here will still work and be useful.]*

When we compiled the subroutine and attempted to call it from a BASIC program, we encountered a difficulty. The Microsoft compiler was not designed for compiling subroutines which are to be called from a BASIC program. Unfortunately, the execution of the compiled routine erases many of the pointers used by the main program. Hence, these pointers need to be preserved for later recall in order to return to the calling program. We include here our solution to this problem with the Microsoft compiler in the hope that you can incorporate it directly or find a way to modify it for your own use. In our case, the results were well worth the effort.

Two interface routines were written; one routine creates a path from the BASIC program to the compiled subroutine and the second creates a path from the compiled program back to the BASIC program. These are presented in Listing 1.



The routine ENTRY performs the following functions:

1. Retrieves from the stack the return address in the BASIC interpreter and saves it for use by routine EXIT;
2. Stores the contents of all 256 page zero locations;
3. Transfers control to the compiled subroutine.

Routine ENTRY is called from the BASIC program. During execution of the compiled program, the contents of several page zero locations are altered. Consequently, the original contents of these locations have to be saved in order to resume execution of the BASIC program following the call to the compiled subroutine.

The routine EXIT performs the following functions:

1. Restores to the top of the stack the return address in the BASIC interpreter saved by routine ENTRY;
2. Restores the contents of the page zero locations;
3. Issues a 'return from subroutine' command.

Routine EXIT is called from the compiled subroutine. The functions it performs are necessary in order to allow execution of the BASIC program to resume at the statement following the call to the routine ENTRY.

We stored these routines in the first part of page three in memory which is available for user programs. A page of memory is also needed to save the contents of page zero prior to execution of the compiled program. Since our programs did not open any disk data files, we used a page ($96) allocated to DOS as a file buffer. Obviously, any unused memory page could be used.

Figure 1 illustrates the use of the ENTRY and EXIT routines. Both routines have been stored in a disk file called **ENTRY-EXIT ROUTINES.OBJ**. Note that the addresses appearing in the assembler Listing 1 are given in base 16, whereas the addresses used in the programs below are in base 10. (Note: $300_{16} = 768_{10}$ and $31A_{16} = 794_{10}$.)

One should note that the last line of routine ENTRY of the assembler listing is a jump to the compiled subroutine. In this example, the address is $684C. This address will change depending upon where you wish to store this routine in memory. If your BASIC program calls more than one compiled subroutine, the same ENTRY and EXIT routines can be used by changing the address portion of the JMP statement before the call to the ENTRY routine. This can be done using the BASIC POKE statement.

Consider the following example. We are assuming that the first compiled subroutine is stored at $1040 and the second is stored at $6200. Note that the address in the JMP instruction is stored low-byte followed by high-byte. (Note: $10_{16} = 16_{10}$, $40_{16} = 64_{10}$ and $62_{16} = 98_{10}$.)

Figure 1

```
MAIN PROGRAM
  .
  .
  .
400 PRINT CHR$(4)+"BLOAD ENTRY-EXIT ROUTINES.OBJ"
  .
  .
  .
2000 CALL 768 : REM CALL ENTRY ROUTINE
  .
  .
  .
SUBROUTINE TO BE COMPILED
  .
  .
  .
CALL 794 : REM CALL EXIT ROUTINE
```

Figure 2

```
MAIN PROGRAM
  .
  .
  .
400 PRINT CHR$(4)+"BLOAD ENTRY-EXIT ROUTINES.OBJ"
  .
  .
  .
2000 REM CALL FIRST COMPILED SUBROUTINE
2010 POKE 792,64 : REM STORE LOW-BYTE OF ADDRESS
2020 POKE 793,16 : REM STORE HIGH-BYTE OF ADDRESS
2030 CALL 768    : REM CALL ENTRY ROUTINE
  .
  .
4000 REM CALL SECOND COMPILED SUBROUTINE
4010 POKE 792,00 : REM STORE LOW-BYTE OF ADDRESS
4020 POKE 793,98 : REM STORE HIGH-BYTE OF ADDRESS
4030 CALL 768    : REM CALL ENTRY ROUTINE
  .
  .
  .
```

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!**

☐ **MICRO on the APPLE — Vol. 3** A $24.95 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) This offer good in USA only

I'm paying by: ☐ Check ☐ MO ☐ VISA ☐ MC

Total Enclosed:$ ____________

Visa/MC # ____________

Exp. Date: ____________

APRIL 15 REPEATED

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 — Now Only $14.95

Please rush the MICRO Books I've checked above to:

Name ______________________

Address ______________________

City ____________ State ________ Zip ________

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

I'm paying by: ☐ Check ☐ MO ☐ VISA ☐ MC

Total Enclosed: $ ____________
(Add $2.00 s/h per book.)

Visa/MC # ____________

Exp. Date: ____________

MICRO™'s Best Sellers

The Best of MICRO

Make your computer reference library complete by adding the "Best of MICRO" to your present book collection. Each volume full of articles and programs pulled from the pages of MICRO. Excellent software at a very low price!

☐ BOM Volume 2 @ $8.00
☐ BOM Volume 3 @ $10.00

What's Where in the Apple

Revised new edition with Apple IIe information added to 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 **@ $19.95**

Best Sellers for APPLE Users!
MICRO on the APPLE

Programming aids, utilities, games, enhancements. Together Volumes 1, 2, and 3 contain over 100 programs on diskette. Fully documented and explained.

☐ 3-Volume Gift-Boxed @ $59.95
☐ Vol.2 ☐ Vol.3 $24.95 ea.

Please rush the MICRO Books I've checked above to:

Name ______________________

Address ______________________

City ____________ State ________ Zip ________

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

I'm paying by: ☐ Check ☐ MO ☐ VISA ☐ MC

Total Enclosed: $ ____________
(Add $2.00 s/h per book.)

Visa/MC # ____________

Exp. Date: ____________

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY CARD

FIRST CLASS PERMIT NO. 60, CHELMSFORD, MA

POSTAGE WILL BE PAID BY ADDRESSEE

MICRO™

P.O. Box 6502
Chelmsford, MA 01824

SAVE 20%!!!!

Subscribe to MICRO™

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

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY CARD

FIRST CLASS PERMIT NO. 60, CHELMSFORD, MA

POSTAGE WILL BE PAID BY ADDRESSEE

MICRO™

P.O. Box 6502
Chelmsford, MA 01824

A Feast Of Computing Ideas...

New Books From MICRO™

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY CARD

FIRST CLASS PERMIT NO. 60, CHELMSFORD, MA

POSTAGE WILL BE PAID BY ADDRESSEE

MICRO™

P.O. Box 6502
Chelmsford, MA 01824

Order These Best Sellers From MICRO™

Listing 1

```
                  ; ENTRY-EXIT ROUTINE
                  ;
0300                      ORG $0300
                  ;
9600              FREE    EQU $9600      ; FREE PAGE OF MEMORY
684C              SUBR    EQU $684C      ; ADDRESS OF COMPILED SUBR
                  ;
0300 4C 05 03     ENTRY   JMP AROUND
0303              SADDR   DFS 2
0305 68           AROUND  PLA            ; REMOVE ADDRESS FROM STACK
0306 8D 03 03             STA SADDR      ; SAVE FOR EXIT ROUTINE
0309 68                   PLA
030A 8D 04 03             STA SADDR+1
030D A2 00                LDX #$00       ; INITIALIZE LOOP COUNTER
030F B5 00        SAVE    LDA $00,X      ; SAVE CONTENTS OF PAGE ZERO
0311 9D 00 96             STA FREE,X     ; "FREE" PAGE OF MEMORY
0314 E8                   INX
0315 D0 F8                BNE SAVE
0317 4C 4C 68             JMP SUBR       ; JUMP TO COMPILED SUBROUTINE
                  ;
031A 68           EXIT    PLA            ; REMOVE ADDRESS FROM STACK
031B 68                   PLA            ; THIS ADDRESS IS NOT NEEDED
031C AD 04 03             LDA SADDR+1    ; RESTORE RETURN ADDRESS
031F 48                   PHA            ; TO TOP OF STACK
0320 AD 03 03             LDA SADDR
0323 48                   PHA
0324 A2 00                LDX #$00       ; INITIALIZE LOOP COUNTER
0326 BD 00 96     RET     LDA FREE,X     ; RESTORE PAGE ZERO
0329 95 00                STA $00,X
032B E8                   INX
032C D0 F8                BNE RET
032E 60                   RTS            ; RETURN TO BASIC INTERPRETER
                  ;
032F                      END
```

MICRO™

Microbes

It has come to our attention that there were a number of errors in the previously published Investor program by Joseph Kattan. We have traced these problems to a new transmission system. Most of the errors are listed below, there may be a few more but we were able to run the program using the listing given (with the corrections). Our apologies, the problem will be resolved in future listings.

Line 180 at the Return statement is the start of line 185.
Line 256 the line wraps around - GOTO 241 - then line 260 begins.
Line 320 should be - GOTO 315, not 15.
Line 345, second line, line 345 begins.
Line 347, second line, line 350 begins.
Line 425 the '$' is missing - (LEN(N$)+1).
Line 450 the = sign is missing from L=1-1.
Line 1000 the E is missing from REM.
Line 1020 the I is missing from IF.
Line 1110, second line, line 1120 begins.
Line 1210 the G is missing from the GOTO.
Line 1230 the word 'their' should be 'other'.
Line 2500 the missing number should read MI=F(1).
Line 2520 the missing number should read MX=3.
Line 3010 the E is missing from DATE$ at the end of the line.
Line 3340 the = is missing from THEN N2=N1.
Line 4010, second line, line 4015 begins, the GOTO before references line 2000.
Line 5005 the '(' is missing from S(8) at the end of the line.
Line 5100 the '(' is missing from (3) RATE OF RETURN.

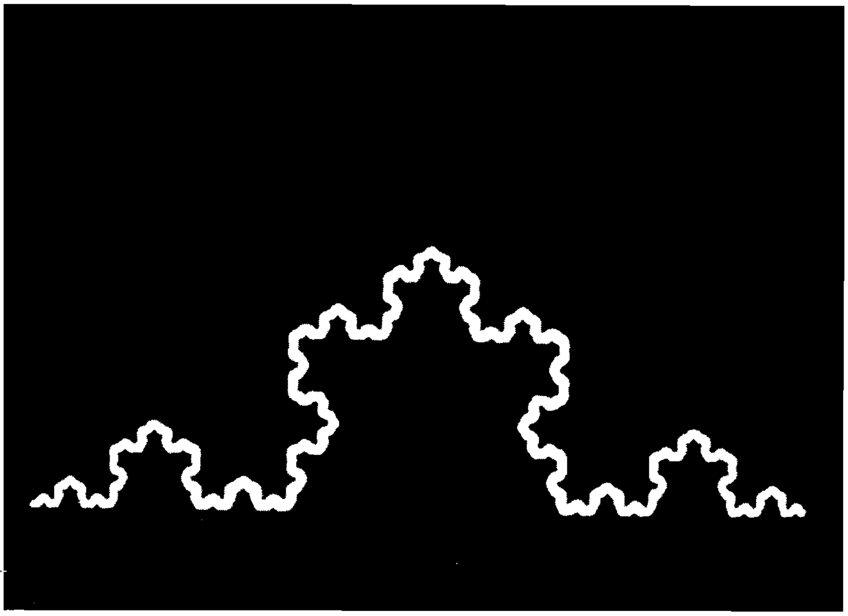


Figure 1

Plotting Fractals On Your Computer

by Simon Wardrop

Plotting fractals (irregular shapes) can often produce beautiful, even spectacular results, but they call into question our common definition of length.

What is a Fractal?

Briefly speaking, a fractal is a plane shape in which the usual notions of length and area cease to be useful. The term was first coined by a French mathematician, by the name of Bernoit Mendelbrot, in his book, "Fractals: Form Chance and Dimension" (W.H. Freeman, San Francisco, 1977); it is derived from the Latin word for "irregular" or "fragmented", which aptly describes their typical appearance (see Figure 1 for example). A really rigorous definition of a fractal would require a long digression into a lot of avoidable mathematics. However, it is not difficult to write a program, for a computer equipped with reasonably high resolution graphics, to create them. Apart from being an interesting exercise in recursive programming (that is, programming in which "Stakko" or "LIFO data structures" are employed), doing this is worth the effort, as the results are often beautiful and spectacular.

The "Snowflake Curve"

One of the simplest fractals is the so-called "snowflake curve" or more technically, "the triadic Koch island" which was discovered early this century by H. von Koch. This shape is produced in the following way: begin with an equilateral triangle, divide each side into three, then replace each middle segment by a smaller equilateral triangle. Then repeat the process on each new segment so produced. The first few stages of this construction are shown in Figure 2. Actually, the program described in this article draws only one third of the snowflake curve. This decision was made because of the limited resolution available on computers. In order to draw the entire snowflake a reduction in size would have been necessary, and so fewer "generations" could have been produced; I opted for a closer look at just one side.

The Program

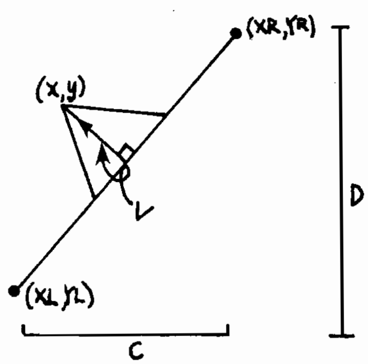
The complete program is shown in Listing 1. I do not think that the program is easy to follow, and neither could I make it so. The process of building the snowflake is fundamentally recursive and BASIC does not handle such processes well. I had no choice in the language used, and BASIC is still the most commonly available language.

What the program does is this. Firstly (line 40) the width and height of the screen (SW,SH) are specified, and then the minimum length (ML) of a line segment in the final curve is given. (This latter specification is necessary to prevent the process from going on forever). Then the starting and ending points of the baseline are pushed onto the stacks x(p), y(p) which operate in "parallel." A subroutine is then called which splits the line specified by the topmost, and second topmost, entries of the stacks into three, and then builds a triangle. The resultant four line segments then have their endpoints pushed onto the stack, according to the scheme shown in Figure 3. (Notice that, at any stage, the endpoint which is closest along the curve from the left is at the top of the stack, while the next closest is next on the stack.) We then return to the main loop where the length of the new segments (a,b,c,d) are compared with the prescribed minimum: if they are smaller, they are popped from the stack and drawn (line 100), otherwise the subroutine is called again.

The most obscure point of the subroutine is line 540. The purpose of this line of code is to ensure that, at

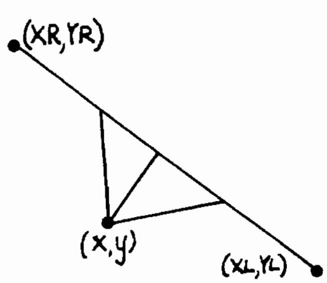
Figure 4

When the point (XL,YL) is actually rightmost on the screen, we still want the triangle sticking out, so:



x = XL + C/2 - Lsin(grad)
y = YL + D/2 + Lcas(grad)

The code that produces the new triangle, on the line segment, (lines 540-580) assumes that the coordinates on the top of the stack, are not only leftmost along the curve, but leftmost on the screen. Thus a typical situation is:



x = XL + C/2 + Lsin(grad)
y = YL + D/2 - Lcas(grad)

Thus, when XL is bigger than XR, we must replace L by -L. Consequently we have line 540 of the program.

Figure 2

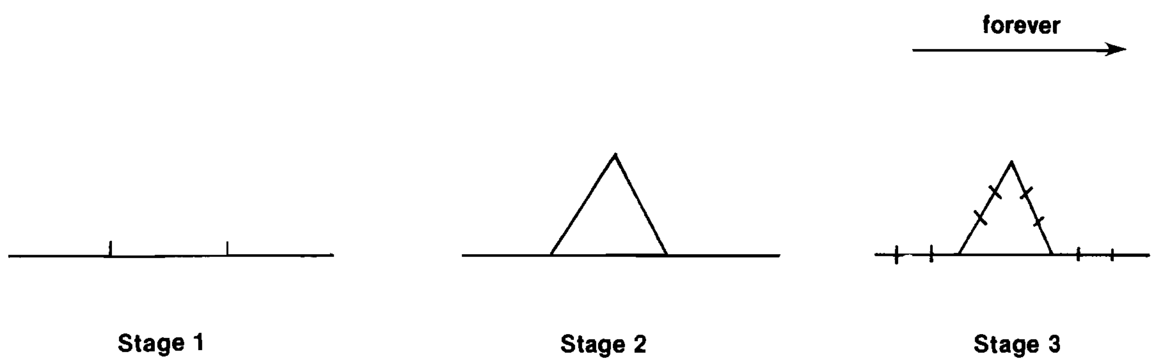


Figure 3

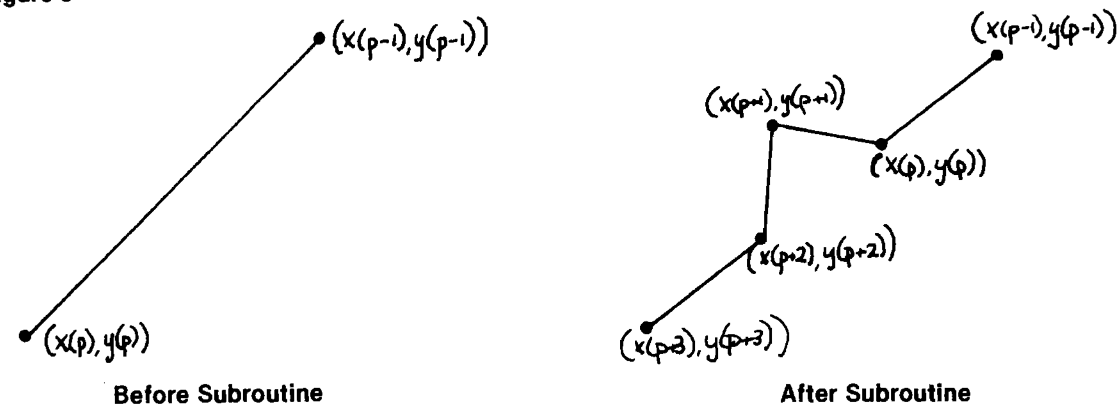
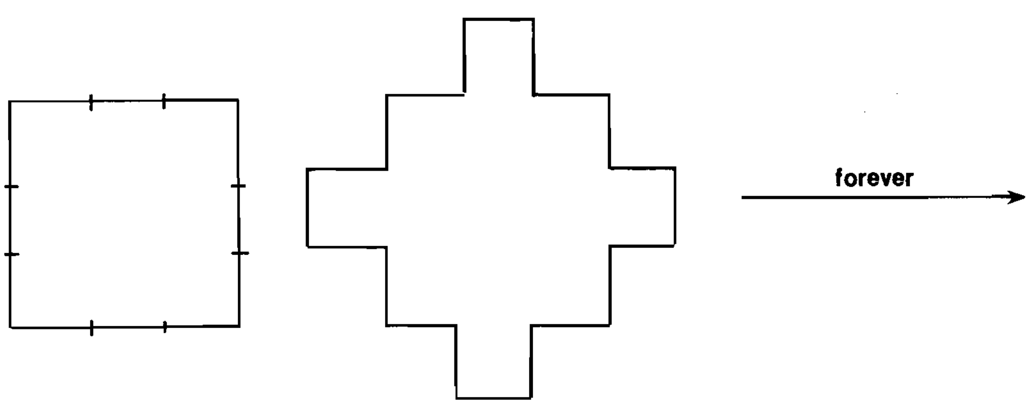


Figure 5



Listing 1

```
  9 REM ****** CLEAR SCREEN COMMAND IS MACHINE SPECIFIC **************
 10 CLS : REM CLEAR SCREEN
 11 REM ****** PROVIDE THE APPROPRIATE CODE FOR YOUR COMPUTER ********
 20 DIM X(1000),Y(1000)
 30 REM ################## INITIALIZE VARIABLES ######################
 40 ML=10:S=.5:P=3.1416:SW=639:SH=199:CZ=COS(PI/6)/3
 50 X(1)=639:Y(1)=10
 60 X(2)=  0:Y(2)=10
 70 P=2
 80 REM ################## BEGIN MAIN LOOP ###########################
 90 L=SQR((X(P)-X(P-1))^2+(Y(P)-Y(P-1))^2)
 99 REM ******* THE NEXT LINE IS MACHINE SPECIFIC FOR PLOTTING ******
100 IF L<ML THEN LINE (X(P),SH-S*Y(P))-(X(P-1),SH-S*Y(P-1)),PSET:P=P-1
101 REM ******* PROVIDE THE APPROPRIATE CODE FOR YOUR COMPUTER *******
110 IF L>ML THEN GOSUB 500
120 IF P>1 THEN GOTO 90: REM WHILE STACK NOT EMPTY, REPEAT PROCEDURE
130 STOP
140 REM ################# END MAIN LOOP ##############################
500 REM ################# SPLIT THE INTERVAL #########################
510 XL=X(P)                    :YL=Y(P)    :XR=X(P-1)    :YR=Y(P-1)
520 C=XR-XL                    :D=YR-YL    :L=SQR(C*C+D*D)*CZ
530 GD=ATN(D/C)
540 IF XR<XL THE L=-L
550 X(P+3)=XL                  :Y(P+3)=YL
560 X(P+2)=C/3+XL              :Y(P+2)=D/3+YL
570 X(P+1)=C/2-L*SIN(GD)+XL    :Y(P+1)=D/2+L*COS(GD)+YL
580 X(P+0)=2*C/3+XL            :Y(P+0)=2*D/3+YL
590 X(P-1)=XR                  :Y(P-1)=YR
600 P=P+3
610 RETURN
620 REM ################# END OF SUBROUTINE ##########################
```

each splitting, the new triangles actually do stick "outwards." The logic behind this is shown in Figure 4.

The program was written for an Hitachi Peach; however, it should run on most machines with good graphics. The only non-standard BASIC used are the statements LINE (X1,Y1)-(X2,Y2), PSET which draws a line from (X1,Y1) to (X2,Y2) and CLS which clears the screen.

There are several extensions to the program that could be made. One could draw all three sides of the curve, and, of course, Peach and Color Computer owners can PAINT under the curve. To improve the speed, one might remove line 590 and the " + 0" in line 580; they were included for clarity

There are many other related "recursively defined" curves that can be produced by the same algorithm as that used in the program. Some examples are shown in Figure 5. For many other interesting Fractals, use Bernoit Mandelbrot's book; the first half of it is fairly easy reading, and there are hundreds of good, computer drawn, plates.

Some Properties of Fractals

There are several interesting features of the snowflake curve that are worth mentioning. Firstly, it has an infinite circumference, but a finite area (a finite area because you can enclose it in a finite square, and an infinite length because, at each stage, you replace the three intervals by four of the same length, and so increase the circumference by a factor of 4/3; so after infinitely many generations the "coastline" of the 'island' is infinitely long).

This is not such an esoteric property. It has been argued that actual coastlines behave similarly. If you measure the length of Australia's coastline with a 1 km "yardstick", you will get a one figure, while if you measure it with a 1 metre "yardstick" you will get another result which is probably bigger than the first. Thus, as you use smaller and smaller yardsticks (and supposedly get increasingly more accurate results), you get bigger and bigger results. What then is the length? It would seem to be infinite. This is the crux of Mandelbrot's book. He argued for a new definition of length that would, hopefully, be more useful for the comparison of "lengths" and "sizes" than the current scheme in which every coastline has the same length: infinity!

Conclusion

I hope that this article has sparked some interest in the fascinating curves called "fractals", and demonstrated a use for stacks other than sorting! There are many possibilities for experimentation. Mathematically the field is far from dead; fractals are being applied to things as commonplace as soap bubbles, and as esoteric as the "strange attractors" of differential equations.

Simon Wardrop may be reached at 3 Gwenda Avenue, Blackburn, 3130, Australia.

From Here to Atari

by Paul S. Swanson

I have been testing my ATR8000 (a peripheral discussed previously in Micro No. 68) in different modes this month and am impressed by its versatility. I recently acquired the latest release of MYDOS which greatly expands the capacity of the ATR8000 when used as an Atari peripheral. MYDOS replaces the functions of Atari DOS and adds others required to support more features of the ATR8000.

MYDOS will act as a direct replacement for Atari DOS when used with most programs. The only exception I found is the result of a bug in MYDOS not allowing random access updating in files, but have received word that the problem is being addressed by the author of MYDOS, Charles W. Marslett. MYDOS is distributed by SWP Microcomputer Products, Inc., in Arlington, Texas 76011, which is the company producing the ATR8000.

Most of the additional features of MYDOS concern the different configurations of disk drives available through the ATR8000. Supported in the 5-1/4 line are single sided single density 40 track disks, like the 810 drives use, giving about 90K of storage, to double sided double density 80 track drives which store over 700K. With a $19.95 adapter for each, the ATR8000 will also support 8 drives. Single sided 8 drives will hold almost 500K and double sided 8 drives will hold 990K. The net result of this is a lot more storage capacity for the Atari computer. For example, using four 5-1/4 double sided double density 80 track drives yields about 2.8 megabytes of on-line storage. Four double sided 8 drives would hold almost 4 megabytes. To transfer back and forth from the single sided single density Atari-compatible disk formats there must be one such disk on line, but for maximum storage that drive can first be used to make the transfers, then removed and replaced with a larger capacity drive for actual operation.

On my system (until I go get a double sided 80 track 5-1/4 drive, anyway) I have two TRS-80 drives connected by a Radio Shack drive cable. Used in double density mode this gives me the equivalent of four Atari disk drives, or about 360K. MYDOS will allow me to define these disks as either single or double density and will automatically redefine them if, for example, I put a single density disk in a double density configured drive, or vice versa. This automatic redefinition will work in DOS mode, but is not automatic once a program is running, so before executing a program it is important to verify that the disks are configured the way you want to use them.

Other interesting features in MYDOS includes modification to the DOS C (copy file), I (initialize) and J (duplicate) commands. Copy may be done from any filespec to any filespec, and with the RS232 version of MYDOS, version 3.16, a file may even be copied from disk to the RS232 port, which is not possible under Atari DOS or without the ATR8000 RS232 port. Initialized, used in Atari DOS as a format command, can be specified to format or just erase the diskette. The duplicate command has two options. First, only a certain range of sectors may be specified. Second, the destination drive may be either formatted or erased before the duplication is done.

Another interesting feature of MYDOS is the ability to create multiple directories on one disk. This eliminates the 64-file limit imposed under Atari DOS. Each additional directory is installed as the equivalent of one file name in the main directory and may also contain up to 64 file names. Since the ATR8000 supports so many different drive configurations, some further control is also supported. For example, different disk drives will respond at different speeds. One of the additional controls sets the amount of time required to move the disk read/write head from one track to the next to accommodate slower drives. This can be set differently for each drive on the system. The default for 5-1/4 disks is 6ms per track, but that may be redefined to as slow as 30 ms per track. Basically, if you are having problems reading disks on a particular drive, you can try slowing it down on the assumption that you are not giving the disk sufficient time to position the head before a read or write operation.

There are also several other double density disks compatible with the Atari computers. MYDOS will support these drives also. Some of these third party disk drives, like the Trak drive produced by Trak Microcomputer Corporation at 1511 Ogden Ave., Downers Grove, Ill. 60515, retailing at $499.00, also has a printer port with a small built-in printer buffer (4K).

MYDOS also has added file manager routines, accessible in BASIC using the XIO command, to support these new functions. On some, the same XIO commands are available as in Atari DOS, but the AUX1 and AUX2 bytes, normally zero for Atari DOS, have some other information in them. These control the type of formatting to be done, the number of sectors on the drive and so forth. XIO commands added include things like creation of new directories and setting the default directory.

In addition, MYDOS 1.16 supports the ATR8000 RS232 port. The basic difference between this RS232 handler and the one used with the 850 interface is that MYDOS contains the handler as an integral part of DOS. This means that, if you are using DUP.SYS (DOS command from BASIC) you don't lose the RS232 handler. It is simply always there. This drawback to the 850 handler is particularly annoying when working in machine language because it is always required to append the object file to the AUTORUN.SYS file used to load the handler. In MYDOS, the machine language routine that accesses the RS232 port can be run directly without worrying about loading any handler.

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

MICRO

MICRO Program Listing Conventions

Commodore

LISTING	C64 KEYBOARD
Commands	
{CLEAR}	^ CLR
{HOME}	HOME
{INSERT}	^ INST
{DOWN}	CRSR DOWN
{UP}	^ CRSR UP
{RIGHT}	CRSR RIGHT
{LEFT}	^ CRSR LEFT

Colors

{BLACK}	CTRL 1 BLK
{WHITE}	CTRL 2 WHT
{RED}	CTRL 3 RED
{CYN}	CTRL 4 CYN
{PURPLE}	CTRL 5 PUR
{GREEN}	CTRL 6 GRN
{BLUE}	CTRL 7 BLU
{YELLOW}	CTRL 8 YEL
{RVS}	CTRL 9 RVS ON
{RVSOFF}	CTRL 0 RVS OFF
{ORANGE}	= 1
{BROWN}	= 2
{GREY 1}	= 3
{GREY 1}	= 4
{GREY 2}	= 5
{LT GREEN}	= 6
{LT BLUE}	= 7
{GREY 3}	= 8

Functions

{F1}	f1
{F2}	^ f2
{F3}	f3
{F4}	^ f4
{F5}	f5
{F6}	^ f6
{F7}	f7
{F8}	^ f8

Special Characters

{PI}	π ^ Pi Char
{POUND}	£ Pound Sign
{UP ARROW}	↑ Up Arrow
{BACK ARROW}	← Back Arrow

Atari

Conventions used in ATARI Listings.

Normal Alphanumeric appear as UPPER CASE:
SAMPLE
Reversed Alphanumeric appear as lower case:
yES (y is reversed)
Special Control Characters in quotes appear as:
{command} as follows:

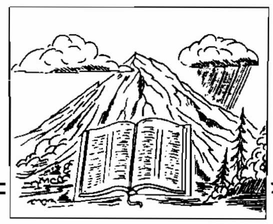
Listing	Command	ATARI Keys
{UP}	Cursor Up	↑ ESC/CTRL -
{DOWN}	Cursor Down	↓ ESC/CTRL =
{LEFT}	Cursor Left	← ESC/CTRL +
{RIGHT}	Cursor Right	→ ESC/CTRL *
{CLEAR}	Clear Screen	↖ ESC/CLEAR
{BACK}	Back Space	ESC/BACK S
{TAB}	Cursor to Tab	ESC/TAB
{DELETE LINE}	Delete Line	ESC/SHIFT DELETE
{INSERT LINE}	Insert Line	ESC/SHIFT INSERT
{CLEAR TAB}	Clear Tab Stop	ESC/CTRL TAB
{SET TAB}	Set Tab Stop	ESC/SHIFT TAB
{BEEP}	Beep Speaker	ESC/CTRL 2
{DELETE}	Delete Char.	ESC/CTRL BACK S
{INSERT}	Insert Char.	ESC/CTRL INSERT
{CTRL A}	Graphic Char.	CTRL A
	where A is any Graphic Letter Key	

Non-Keyboard Commands

{DIS=}	CHR$(8)
{ENB=}	CHR$(9)
{LOWER CASE}	CHR$(14)
{UPPER CASE}	CHR$(142)
{^RETURN}	CHR$(142)
{DEL}	CHR$(20)
{SPACE}	CHR$(160)

Notes:

1. ^ represents SHIFT KEY
2. = represents Commodore key in lower left corner of keyboard
3. CTRL represents CTRL key
4. Graphics characters represented in Listing by keystrokes required to generate the character
5. A number directly after a {SYMBOL} indicates multiples of the SYMBOL: {DOWN6} would mean DOWN 6 times



Advertiser's Index

National Advertising Representatives

East Coast:
John Gancarz
P.O. Box 6502
Chelmsford, MA 01824 (617) 256-3649

Mid-West:
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.

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)