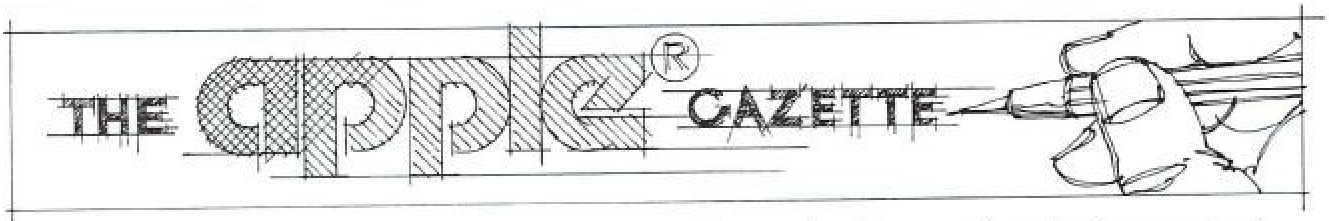


Corvus 11A Disc Drive

A User's Review

by
Michael Tulloch
103 White Cr.
Niceville, Fla. 32578

Manufacturer
Corvus Systems, Inc.
900 S. Winchester Boulevard
San Jose, California 95128
408 725-0920

Would you spend 4 or 5 times more for a disk than you spent for your Apple computer? That's right, $5,300 for an Apple compatible disk. It may sound like a lot of money, but it might be money well spent if you've got large data files or many programs.

The Computer Store of Destin, Florida recently received their first Corvus 11A disc drive. The owner liked it so much that he took it home!

I've used the unit but most of the following information is based upon interviewing Gary Workman, the store owner. I'll tell you what the Corvus disc is, some of its physical specifications and our experience with the unit. If you run a business or do large data reduction programming, you'll find it worth investigating.

The disc unit itself is a space age enclosure of black metal and dark plexiglas. A round wire cable connects the drive to its power supply. A ribbon cable connects the drive to an interface board in the Apple. The interface board plugs into a slot just like most other Apple peripherals. Since the Corvus only occupies one slot, the Apple can still accommodate floppy discs.

High speed is another positive feature of the Corvus. I couldn't accurately measure the difference between the Corvus and the floppy, but I'd guess the Corvus is more than ten times faster in both read and write. This means that serial or random access searches of large data bases become really practical.

One of the most impressive of the Corvus disc features is its capacity. It is equivalent to 82 standard Apple discs: 9.6 megabytes are on line. Each of the 82 volumes are identical in size to an Apple floppy diskette. What is almost as impressive is that this capacity is packed into two moderate sized boxes. One contains the drive (an IMI 7710 "Winchester" hard disc) and drive electronics. It measures 8¼ x 6 x 18 inches. The other box is the power supply which is 4 x 5¼ x 15½ inches.

Documentation is, as usual in new computer products, a weak point. Eight pages, including the limited warranty, were all that came with the unit. Gary was able to get the thing up and running but a couple of calls to the manufacturer were required. Newer units are supposed to come with more complete documentation. In any case, be sure you're familiar with Apple's 3.2 DOS before you jump in. You can't hurt the disc but you sure can hurt your data.

On the positive side of the documentation issue is the excellent telephone help and complete compatability with Apple software.

Another nice feature is the aforementioned complete compatibility with Apple DOS. The Corvus is completely transparent to the user. The volume number in the Apple I/O commands selects the disc volume just like the slot number does with multiple floppy drives. The only software difference is a new command, "CATALOG V99" lists the first program name from each volume. Auto boot is also available.

Gary's Corvus disc has been in use for two weeks with no operating problems. The device incorporates several safety mechanisms so that "head crashes" are apparently a thing of the past. An auto-write-stop feature on power loss is just one of these.

When this unit was ordered they were quoting 2-3 weeks for delivery. Gary's took a month. Delivery now looks like two weeks. This may change as their sales increase.

My only strong dislike is the power switch. It is located on the power supply. The power supply is massive and looks out of place among all the other sleek computer parts. It's unfortunate the power supply can't be hidden somewhere.

The Corvus seems an ideal mass memory device for a business, professional, or other interest requiring large information storage. Its price is steep for strictly hobby use. But for those who can afford it, or need it, it really helps make a small computer into The Big Apple.

APPLE RESOURCES

Apple Computer, Inc.
10260 Bandley Drive
Cupertino, CA 95051

Compute.
Apple Coordinator
900 Spring Garden St.
Greensboro, NC 27403

Mountain Hardware
300 Harvey West Blvd.
Santa Cruz, CA 95060

Personal Software
592 Weddell Drive
Sunnyvale, CA 94086

Automated Simulations
Dept. J
P.O. Box 4232
Mountain View, CA 94040

Programma
3400 Wilshire Blvd.
Los Angeles, CA 90010

Rainbow Computing, Inc.
Garden Plaza Shopping Center
9719 Reseda Blvd.
Northridge, CA 91324

SSM
2116 Walsh Ave.
Santa Clara, CA 95050

Sub Logic
Box V
Savoy, IL 61874

HAYDEN'S APPLECART IS OVERFLOWING WITH...

SARGON II (Spracklens)
"Buy this program when it becomes available--...an evaluation routine that enabled it to beat the giants!...unequaled in the end game..." *Personal Computing, July, 1979.* It is able to push passed pawns toward queening, play a strong end game, and range in deep play levels at end game without user direction. Has 7 levels of play and levels 0-3 play in tournament time. It has a randomized opening book for all 7 levels of play through 3 moves. A hint mode is included at all levels of play but 0. *#03404, Apple II, $29.95.*

SLOW LIST/STOP LIST (Gilder)
Start, stop, and control the speed of your program with Apple II's game paddles. You can also control the speed at which the disk catalog appears and even terminate the CATALOG operation in the middle. The program can also be enabled and disabled under software control *#03904, Apple II, $10.95.*

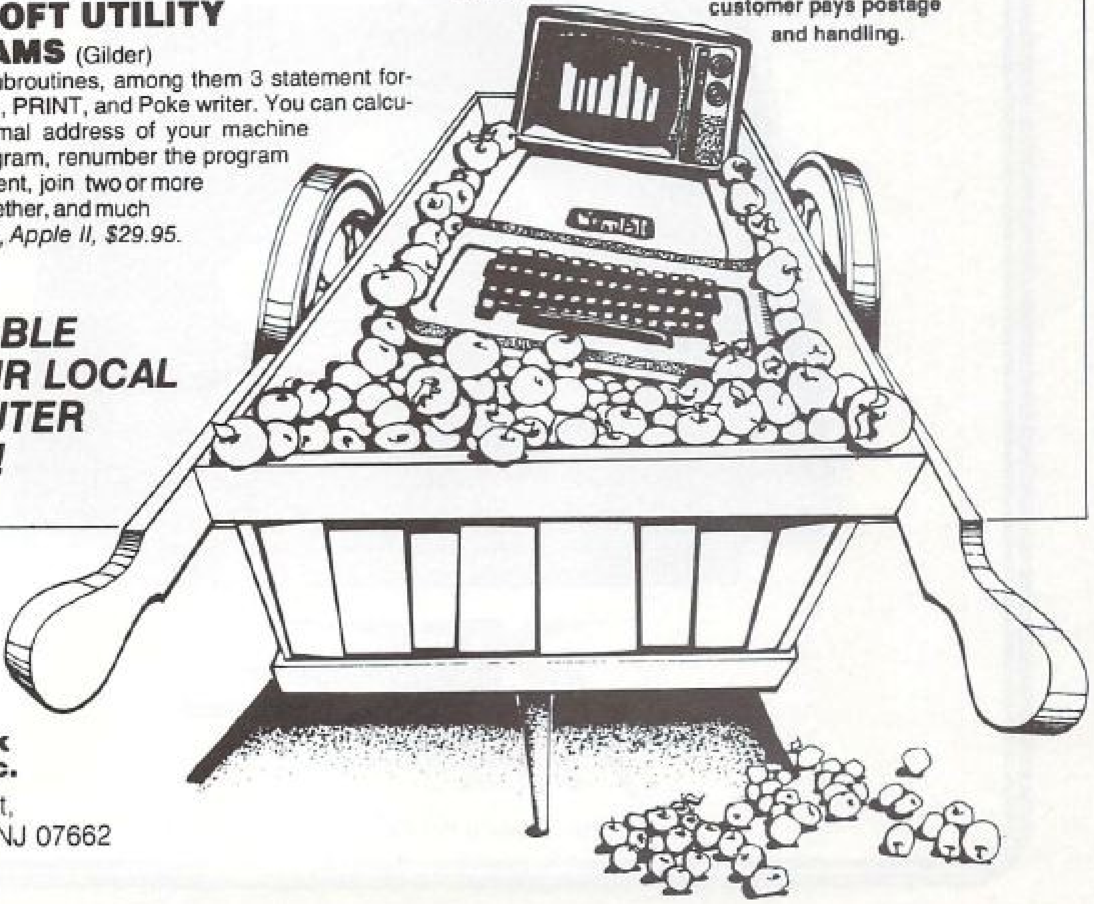
APPLESOFT UTILITY PROGRAMS (Gilder)
Contains 9 subroutines, among them 3 statement formatters: REM, PRINT, and Poke writer. You can calculate the decimal address of your machine language program, renumber the program in any increment, join two or more programs together, and much more. *#03504, Apple II, $29.95.*

AVAILABLE AT YOUR LOCAL COMPUTER STORE!

REVIVE (Gilder)
When a program is accidentally erased, *REVIVE* searches through memory and finds the information that enables it to restore the pointers that have been changed. Can be loaded at any time, before or after you have entered an Applesoft program, or before or after you have accidently erased the program. *#03604, Apple II, $19.95.*

SONGS IN THE KEY OF APPLE
(Lopatin) Allows you to see and hear your favorite tunes, pre-programmed tunes, or music you create (up to 200 notes, including rests per musical piece). Multicolor graphics accompany all music. *#03304, Apple II, $10.95.*

CALL TOLL FREE
800-631-0727, ext. 307
TO CHARGE YOUR ORDER TO Master Charge or BankAmericard! Minimum order is $10.00; customer pays postage and handling.



Hayden Book Company, Inc.
50 Essex Street,
Rochelle Park, NJ 07662

Machine Language Versus Basic Prime Number Generation

Marvin L. De Jong
Dept. of Math-Physics
The School of the Ozarks
Pt. Lookout, MO 65726

Editor's Note:
Watch your local dealer for Marvin L. De Jong's new book: ***Programming and Interfacing the 6502.*** *Due in January, the 450 page work is expected to sell for $11.95. Publisher: Howard W. Sams Co. Indianapolis, IN.*

The attached program will calculate prime numbers of the form $2^N - 1$, for poster and/or prime number enthusiasts who also read COMPUTE. It was motivated by one of my students who was searching for perfect numbers (numbers whose factors add to give the number itself). The student wrote a BASIC program for an APPLE, and the program would calculate and print 2^{10000}. It took 11 hours to do this. Thinking that perhaps the same thing could be done in machine language, I wrote the program given here with only minor modifications. It calculated 2^{10000} in 11 minutes, illustrating the advantage in speed that machine language offers for certain tasks.

The program listed here calculates and prints $2^{11213} - 1$, a number that is known to be prime. With a little more memory space than the 4K on my AIM 65, one could calculate and print the largest known prime (as of this writing) number, namely $2^{44497} - 1$. The number of digits in a number of the form 2^N can be shown to be $1 + N\log 2$. In the program given we calculate 2^{11213} giving the number of digits as $1 + 11213\operatorname{Log}_{10}(2) = 3376$ digits. The number 2^{44497} requires 13395 digits. Each memory location can store two BCD digits, so 2^{11213}, requires 1688 or about 2K locations in memory.

Some notes on the program follow. We allocated locations $0400 to $0FFF to hold the number. This is many more locations than are required to find 2^{11213}, but the program was used to find some larger powers of two also. First, the locations that are to contain the number are cleared to zero. This occurs in instructions $0208 to $0220. The indirect indexed addressing mode is used to reference the memory locations to be cleared. The address of this table is stored in $0004 and $0005. Next, a one is stored in the lowest address of the table. This number is doubled 11213 times giving 2^{11213}. The locations $0000, $0001, and $0002 keep track of the number of doubling times. In the instructions located from $0261 to $0272 this number is tested to see if it has reached 11213. Finally, one is subtracted from the number and it is printed by calling an AIM 65 subroutine at $F000. Owners of other systems can simply use their own output subroutine. It should also be clear from this explanation and the program comments what locations in the program must be modified to handle other numbers of the form 2^N.

There is really no practical use for the program or the output. However, prime numbers and perfect numbers have been of considerable interest to mathematicians for centuries. Perhaps some 6502 user will discover an even larger prime number than 2^{44497}, but don't underestimate the task.

P.S. A lot of leading zeros get printed before the number starts.

```
$0208  A9 00     START  LDA $00         Load pointers to number table.
 020A  85 04            STA TABLE
 020C  A9 04            LDA $04
 020E  85 05            STA TABLE+1
 0210  A0 00            LDY $00         Initialize Y index to zero to
 0212  A9 00     NEXT   LDA $00         clear all table locations to zero.
 0214  91 04     LOOP   STA (TABLE),Y   Put zero in each table location.
 0216  C8               INY             Increment Y to fill page with zeros.
 0217  D0 FB            BNE LOOP
 0219  E6 05            INC TABLE+1     Go to the next page in the table.
 021B  A5 05            LDA TABLE+1     Are all the pages completed?
 021D  C9 10            CMP $10
 021F  90 F1            BCC NEXT        No. Then fill another page.
 0221  A9 04            LDA $04         Yes.  Reset pointers to the base
 0223  85 05            STA TABLE+1     address of the table.
 0225  F8               SED             All subsequent additions will be
 0226  A9 01            LDA $01         in decimal.
 0228  8D 00 04         STA TABLO       Start with one in lowest digit of
 022B  A9 00            LDA $00         the table.
 022D  85 00            STA LO          Initialize the addition counter to
 022F  85 01            STA MID         zero;  three locations ($00,$01,$02)
 0231   85 02                STA HI          in page zero.
 0233   18          COUNT    CLC             Clear carry for additions.
 0234   A9 01                LDA $01         Increment the addition counter,
 0236   65 00                ADC LO          LO, MID, and HI each time the number
 0238   85 00                STA LO          is added to itself.
 023A   A5 01                LDA MID         Carry from LO addition into MID.
 023C   69 00                ADC $00
 023E   85 01                STA MID         Result into MID.
 0240   A5 02                LDA HI          Carry from MID addition into HI.
 0242   69 00                ADC $00
 0244   85 02                STA HI          Result into HI.
 0246   18                   CLC             Clear carry for adding THE NUMBER.
 0247   B1 04       PAGAD    LDA (TABLE),Y   Get a piece of THE NUMBER.
 0249   71 04                ADC (TABLE),Y   Add it to itself.
 024B   91 04                STA (TABLE),Y   Store THE NUMBER.
 024D   C8                   INY             Increment Y to repeat the addition
 024E   D0 F7                BNE PAGAD       for an entire page of memory.
 0250   E6 05                INC TABLE+1     Increment the page number.
 0252   08                   PHP             Store P to keep track of any carry.
 0253   A5 05                LDA TABLE+1     Have we finished adding the entire
 0255   C9 10                CMP $10         table?
 0257   B0 04                BCS DOWN        Yes.  Then check to see if we have
 0259   28                   PLP             added enough times.  No.  Add more.
 025A   4C 47 02             JMP PAGAD
 025D   A9 04       DOWN     LDA $04         Reset table pointer.
 025F   85 05                STA TABLE+1
 0261   A5 00                LDA LO          Check add counter.  Is it equal to
 0263   C9 13                CMP $13         011213?
 0265   D0 CC                BNE COUNT
 0267   A5 01                LDA MID
 0269   C9 12                CMP $12
 026B   D0 C6                BNE COUNT
 026D   A5 02                LDA HI
 026F   C9 01                CMP $01
 0271   90 C0                BCC COUNT
$0273   18                   CLC              Subtract one from 2^11213 to get
 0274   AD 00 04             LDA TABLO        THE PRIME NUMBER.
 0277   E9 00                SBC $00
 0279   8D 00 04             STA TABLO
 027C   A9 0F                LDA $0F          Point to the top of the table to
 027E   85 05                STA TABLE+1      read THE PRIME NUMBER out from
 0280   A0 FF       UP       LDY $FF          the most-significant digit to the
 0282   B1 04       THERE    LDA (TABLE),Y    least-significant digit.
 0284   A2 FE                LDX $FE          Convert the BCD digits to ASCII.
 0286   48                   PHA              Save two digits on the stack.
 0287   4A                   LSR A            Get the most-significant nibble.
 0288   4A                   LSR A            Move it into the low-order nibble.
 0289   4A                   LSR A
 028A   4A                   LSR A
 028B   18          HERE     CLC
 028C   69 30                ADC $30          Here we have an ASCII digit so
 028E   20 00 F0             JSR PRINT        jump to the output routine.
 0291   E8                   INX              Do we need to get another digit?
 0292   F0 06                BEQ AHED         No.
 0294   68                   PLA              Yes.  Get the digits.
 0295   29 0F                AND $0F          Mask the high-order nibble.
 0297   4C 8B 02             JMP HERE         Convert it to ASCII.
 029A   88          AHED     DEY              Get some more of the number from
 029B   C0 FF                CPY $FF          the same page.
 029D   D0 E3                BNE THERE
 029F   C6 05                DEC TABLE+1      Change pages.
 02A1   A5 05                LDA TABLE+1
 02A3   C9 04                CMP $04
 02A5   B0 D9                BCS UP
 02A7   00                   BRK              Back to the monitor.
```

THE apple® GAZETTE

SOFTSELL ASSOCIATES (2022-79th Street, Brooklyn, N.Y. 11214) has announced an Applesoft tape verification program. According to Softsell, a major drawback of the Apple computer is its lack of a verify capability for Applesoft programs that have been saved out on tape. Now, with a new program from Softsell Associates, this problem can be eliminated. Once run, the Applesoft Tape Verifier will provide either an Apple II or an Apple II Plus computer with the ability to verify programs saved to cassette. The program remains resident in the computer as long as power is applied and the computer is in the Applesoft mode.

In addition to working on both types of Apple computers, the Applesoft Tape Verifier also works with both RAM or ROM Applesoft. The program costs $20 and is supplied on an Apple compatible cassette.

Dealer inquiries are invited. ©

Apple Authors

We know you're out there. COMPUTE wants the Apple Gazette to enjoy a long and happy life. We need your input! Send articles, reviews, news items, and club notes to: Apple Coordinator, COMPUTE, Post Office Box 5119, Greensboro, NC 27403 ©

Dear Sir:

We read with interest the review by Michael Tulloch, "D. C. Hayes Micromodem $395.00", and appreciate the favorable review given by him.

The product reviewed was the MICROMODEM II™ which is supplied with our FCC registered MICROCOUPLER™. We also produce a modem for S-100 systems which is called the MICROMODEM 100™. The product names MICROMODEM II, MICROMODEM 100 and MICROCOUPLER are trademarks of D. C. Hayes Associates, Inc. In the future please acknowledge our trademarks as you do for Commodore, Apple and Atair. Also please note that our company name is D. C. Hayes Associates, Inc. and refer to the company by its proper name.

Apple II systems equipped with our MICROMODEM II are being used in an increasing range of applications for business, educational and personal use and we are always glad to hear of new and innovative solutions which involve our products.

I wish you the best with your new publication and look forward to reading the next issue.

Sincerely,
D. C. HAYES ASSOCIATES, INC.
Dennis C. Hayes
President
10 Perimeter Park Drive
Atlanta, Georgia 30341 ©

"Eyes" for Your Apple:

Summagraphics Corporation, a leading manufacturer of Data Tablets/Digitizers, has announced receipt of a large order from Apple Computer, Inc. for Bit Pad One™ Digitizers.

The Bit Pad One Digitizers will be used with Apple II computers for entering graphic data--allowing educators, business people, artists, scientists and others to create circuit designs, original art, and other graphic applications.

For more information, contact Summagraphics, P.O. Box 781, Fairfield, CONN 06430 ©

A Printer for the Apple: The Heath H14

Don Earnhardt
2130 Nettlebrook Drive
Winston-Salem, NC 27106

Most hobbists find it difficult to fit the desired microcomputer system into a budget already suffering from terminal inflation. As a result, one must search for the best prices available and occasionally make compromises in terms of capabilities and features in order to complete the system. Additionally, it is usually necessary to purchase components and peripherals on a piecemeal basis. This provides the ability to spread the cost over a longer time period without finance charges.

With new and at times less costly equipment being introduced almost daily, the piecemeal purchasing plan allows one to choose from a wider variety of components as time passes. However, the question of system compatibility must be the utmost consideration. This question has played a major part in the completion of my own system.

When the time finally arrived for me to purchase a printer for my Apple II® , I had many things to consider. Did I need letter quality? Would I be satisfied with an electrostatic? What would be the minimum acceptable printing speed? Did I need a tractor-feed, friction-feed, or both? Which printer could be readily interfaced with the Apple? And finally, where could I get the most printer for my money?

It was about this time that I began to notice reviews of the Heath H14 in many microcomputer-related magazines. All of these reviews suggested that the H14 kits could possibly be one of the better buys on the market. But would it be comparable? It is advertised as an RS-232 printer acceptable characters in ASCII form and having a data transfer rate of up to 4800 baud. After reading all of the specs in the Heath catalog, I decided that this particular printer could suit my needs - provided it could be successfully interfaced to the Apple II.

Upon placing the order for the H14, the search began for an RS-232 interface. I settled on the serial I/O interface sold for $42.00 by Electronic Systems. P. O. Box 21638, San Jose, CA 95151. The decision was based mainly on the fact that this interface was designed specifically for the Apple. It also came in kit form, thus providing an additional cost advantage.

The RS-232 interface kit arrived in little more than a week and was completed in about 30 minutes. I found that this board was generally well designed. The only shortcoming was the fact that a capacitor in the clock circuit must be soldered, removed, and replaced to change the baud rate. The capacitor supplied with the kit is a .1 MFD which selects a baud rate of 110 (10 characters per second). This is a much slower rate than I had desired, but it is acceptable for purposes of testing the interface and the printer since handshaking is not necessary at this rate.

I was fortunate in that I was able to borrow an RS-232 printer with which to test the interface board and the accompanying software. First, I ran the baud-rate adjustment program which repetitively displays the current baud rate on the video monitor. As the trim-pot on the interface board is adjusted, the display changes to reflect the new baud rate. The baud-rate adjustment program is quite useful, but I was not as pleased with the output program supplied. After writing my own output program, it became clear that the RS-232 interface would work quite well at 110 baud.

After several notices of shipping date postponements, the Heath H14 finally arrived. It was difficult to suppress my anxiety in order to do a thorough job of building, double checking, and testing as recommended in the assembly manual. However, I knew that the recommended construction method could save much time in the long run.

Upon applying power to the printer, I checked all the paper transport features (form-feed, paper advance, and paper reverse). All of them worked as described in the manual. Next, I tested the print mechanism by pressing the test button located on the main circuit board of the printer. The printer zipped across the page, leaving a trail of perfect characters. After verifying that the narrow characters would print correctly, it was time for the most important test.

The H14 was connected to the Apple via the Electronic Systems RS-232 interface. Upon getting into BASIC, a short program was loaded into RAM and the printer output program was activated. I typed the word LIST and H14 began to print the program just as it appeared on the screen. It worked beautifully.

The next step was to increase the baud rate. Three things had to be accomplished in order to do this. The capacitor on the interface board had to be replaced, the new board had to be adjusted, and the handshaking arrangements had to be made.

I selected a 10075 MFD capacitor for 1200 baud, installed it, and utilized the baud-rate adjustment program to make the final adjustments.

ULTIMATE JOYSTICK FOR THE APPLE II

$49.95

The Apple Joystick is a quality crafted dynamic interactive I/O device engineered specifically for the apple computer. The stick comes completely wired for paddles 0 & 1 and switches 0, 1 & 2. Among the excellent features of the stick are auto-centering, which positions the stick in the center of its range whenever the handle is released, and positive action switches with tactile feel and audible feedback.

The stick assembly itself is a precision molded unit originally designed for the ultimate in smooth linear proportional control required for international radio-control model competition.

The heart of the stick centers around two cermet resistive elements with bifurcated wiper contacts, which provide the smooth continuous change in resistance not found in wire-wound elements.

As an added bonus, all game I/O connections are brought out and terminated in the cabinet. This feature facilitates modification and/or implementation of all game I/O functions, such as, (example: annunciators, sound, paddles 2 and 3). Using Gesu's double I/O extender cable and two joysticks (one modified for paddles 2 and 3) two player joystick games can be implemented.

Normally no adjustment is required upon installation of the stick in your Apple computer. However, if it should become necessary to adjust the centering, mechanical adjustment tabs are provided inside the stick cabinet.

Refer to the Apple II reference manual for directions on how to install the stick in your computer.

GAME I/O EXTENDER CABLES

SINGLE $10.00
DOUBLE $16.00

The single model consists of one foot of cable, one 16-pin male and one 16-pin female connector. The extender plugs into the game I/O and the female end if secured to the outside of the cabinet with the double-backed mounting tape provided. Installed in this fashion the extender eliminates the necessity of opening the apple computer to install or remove the stick or any other game device.

The double model is exactly the same as the single model with the addition of a second 16-pin female connector. This extender has the same advantages as the single extender plus allowing two sticks or game I/O devices to be installed simultaneously. Note: When two games I/O devices are installed simultaneously make sure no conflicts exist betwen paddle assignments. Only one device should be assigned to each paddle.

ComputerWorld

6791 WESTMINSTER AVENUE
WESTMINSTER, CA 92683
(714) 891-2587 TELEX 182274

This proved to be no problem, but the handshaking was another matter.

The interface board provides a data terminal ready (DTR) lead which is normally used to indicate that power has been supplied to a modem. The printer provides a request-to-send (RTS) lead that indicates when the printer line buffer is ready to accept characters. Also, the printer sends a CONTROL Q and CONTROL S to the computer via the RS-232 input to provide handshaking capability through software.

I felt that the software handshaking method would be the easiest and less time-consuming. All that was needed was to add a loop in my output program to monitor the status of the RS-232 input port. As long as the control Q was being received (control Q indicates that the printer is busy), it would be safe to continue sending characters. This proved to be correct, and the printer operated very well at 1200 baud.

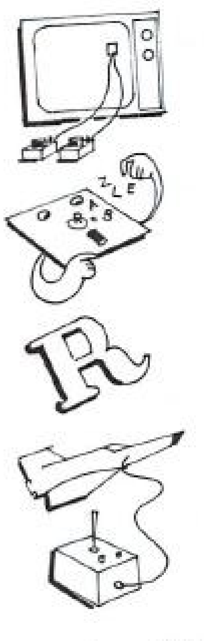
The output program is shown in Figure 1. To link this program to the Apple system software, the address of the START label ($0317) is first entered in the Apple monitor output registers ($36 and $37). Then the first character to be printed is saved on the stack and a JSR to $FDF0 sent to the video is performed to output the character to the Apple video. After returning from this routine, the WAIT1 loop monitors the RS-232 input (for I/O slot 0 on the Apple board, this address is $C080). If a control S (#$13) is present, the printer is indicating that it cannot accept any more characters. If control Q (#$11) is present, the printer buffer is not yet full. But before sending a character, the status of the UART transmit buffer ($C081) must be checked to verify that the UART is ready to accept characters. When the UART is ready, the character previously saved is pulled off the stack and sent to the UART for transmission. The EXIT label provides a return so that another character can be retrieved for processing. When it is desired to turn off the printer, the program resets the monitor output registers to transmit characters only to the video display.

The program can be accessed through the jump table or by going directly to the BON label (30CG from the monitor or call 780 from BASIC). Also the jump table may be used to turn off the printer or the B0FF label may be accessed directly (336G or call 822).

I feel that the H14 is a good choice for Apple owners who are on a tight budget but would like to have a fast dot matrix type printer. Heath Kits and documentation are always good and the fact that troubleshooting information is provided could mean a savings in time and money if trouble should ever occur.

```
                                  .OS
                                  .BA $300
0300- 4C 41 03                    JMP MON
0303- 4C 45 03                    JMP MOFF
0306- 4C 0C 03                    JMP BON
0309- 4C 36 03                    JMP BOFF
030C- A9 17         BON           LDA #$17
030E- 8D 36 00                    STA $36
0311- A9 03                       LDA #$03
0313- 8D 37 00                    STA $37
0316- 60                          RTS
0317- 48            START         PHA
0318- 20 F0 FD                    JSR $FDF0
031B- AD 80 C0      WAIT1         LDA $C080
031E- C9 11                       CMP #$11
0320- D0 F9                       BNE WAIT1
0322- AD 81 C0      WAIT2         LDA $C081
0325- 6A                          ROR A
0326- 90 FA                       BCC WAIT2
0328- 68                          PLA
0329- 8D 82 C0                    STA $C082
032C- C9 8D                       CMP #$8D
032E- D0 05                       BNE EXIT
0330- A9 0A                       LDA #$0A
0332- 4C 17 03                    JMP START
0335- 60            EXIT          RTS
0336- A9 F0         BOFF          LDA #$F0
0338- 8D 36 00                    STA $36
033B- A9 FD                       LDA #$FD
033D- 8D 37 00                    STA $37
0340- 60                          RTS
0341- 20 0C 03      MON           JSR BON
0344- 00                          BRK
0345- 20 36 03      MOFF          JSR BOFF
0348- 00                          BRK
                                  .EN
```

APPLE II SOFTWARE



CURSOR PILOT

gives any Apple II game-paddle control of the video cursor. Activate by touching 'ESC', then edit or copy with game-paddle. Supports normal keyboard controls, is transparent to your programs.

on cassette . . . $5.95

DATA HANDLER

data base management system. Supports infinite data bases on the Apple II disk drive. Structure data to meet your own needs, up to 255 fields per entry. Advanced data processing allows searching and math to generate reports, extensions, and ledgers. Use for inventory, checks, phone numbers, stocks, lab data, etc. Requires 32K & a disk drive.

on diskette with manual . . . $49.95

TYPESETTER

a complete HI-RES graphics character generator and editing system. Allows colors, scaling, upper/lower case, inverse, and can HPLOT letters to any point on the screen. Outputs through regular PRINT statements. Use it to label graphs, create ad displays, or print lower case. System includes 35 utility programs and character sets. When ordering, specify it for disk or ROM Applesoft. Needs 32K with ROM, 48K with disk.

on diskette with manual . . . $24.95

HIRES UTILITY PACK

Why sweat over HI-RES graphics? Shape Generator lets you build graphic shapes with game paddles, see them at all scales, colors, and rotations. Save them to disk, and Shape Adder puts up to 255 shapes together into a table. Utility Subroutines let you position without plotting, find your last plot, and look at the screen to see if a point is on. Requires 16K with Applesoft ROM.

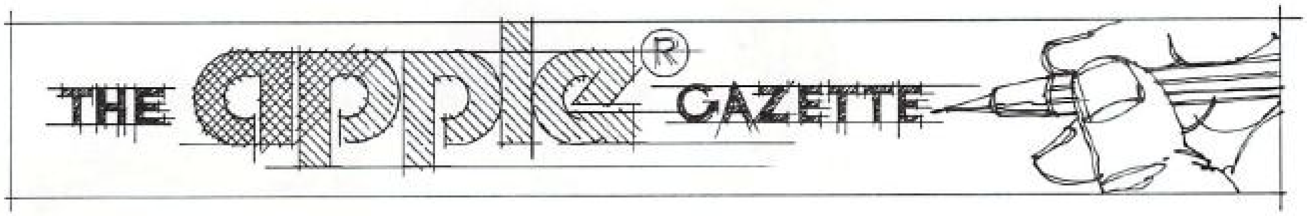
on diskette . . . $14.95

AVAILABLE AT YOUR LOCAL DEALER, OR CALL DIRECTLY AT:

ANDROMEDA COMPUTER SYSTEMS P.O. BOX 19144 GREENSBORO NC 27410 (919)852-1482

Visa and Mastercharge gladly accepted

Apple II and Applesoft are trade marks of the Apple Computer Company, Inc.



NAMING APPLE CASSETTE FILES

D. P. Kemp
1307 Beltram Ct.
Odenton, MD 21113

To owners of disk based Apples the audio cassette may seem like a relic from the dark ages of personal computing, but for many it is still the only form of mass storage available. Unfortunately the Apple's cassette routines are the most rudimentary of those used in 6502-based systems. While the Pet and AIM support named cassette files and the KIM and SYM use numeric file identifiers, the Apple provides no means of file identification. In addition it requires the user to remember and enter memory addresses when loading binary files, and its method of synchronization requires the user to position the tape properly before entering a load command.

The short program listed here was written to alleviate these problems. It writes a descriptive label at the beginning of each file on a cassette and later reads the labels back to enable the location of any particular file. Because it has nothing to do with the actual data in a file the program can be used with any file type including binary, integer, and Applesoft as well as non-standard files such as text editor data, Appleodion scores, or relocatable object files. It can be used to name new files as they are created or to label previously existing files on the cassette.

The program occupies memory from $380 through $3EF and uses the user function (cntl-Y) vector at $3F8. To write a file label, simply type cntl-Y and cntl-W followed by up to 40 characters of labeling information. Start the cassette in record mode, then hit return. The label will be written in about two seconds, after which the normal procedure can be used to save the file. To read labels back, type cntl-Y, cntl-R and return, then start the cassette in play mode. If you are searching for a particular file on a long tape, you may fast forward or rewind to get to its approximate location, then play the tape until the label is found. At this point you should hit any key to exit the read routine, stop the recorder, then load the file as you normally would.

There are three points to be remembered when using this routine:

1. The cntl-Y cntl-W command should be the only one on its line. Any commands appearing before it will displace the label; any appearing after it will be ignored. File labels of more than forty characters may be typed but only the first forty are retained on tape.
2. The cntl-Y cntl-R command should be used only when the cursor is at the bottom of the screen, otherwise scrolling will not occur and subsequent file labels will write over each other on the screen.
3. Cntl-Y followed by anything other than cntl-W or cntl-R will cause a breakpoint.

The label's contents are completely up to the user but I tend to follow a specific convention when labeling my own files. The first character indicates the file type - I use I, A, and B as in Apple's DOS or any other character that seems appropriate for special files. A descriptive filename comes next, followed by any pertinent operating information. Since files are not loaded by name there is no need to minimize typing effort, so abbreviations are a no-no. Example labels are:

I	HIRES KALEIDOSCOPE (NEEDS PROG AID)
B	HIRES SUBS (PROG AID #1) - 6000.63FF
A	DIGITAL FILTER RESPONSE PLOTTER

This routine has eliminated the necessity of storing only one or two files on each side of a short cassette; I would guess that each side of a C-90 is now capable of holding about four disk sides worth of programs and data. The label routine is easy to operate and small enough to remain in memory permanently, and it now occupies the first file of each Apple cassette I own.

Listing 1: Apple cassette file label routine

```
0380-   C8          INY
0381-   B9 00 02    LDA   $0200,Y      ;fetch next character
0384-   C9 92       CMP   #$92         ;cntl-R?
0386-   F0 3F       BEQ   $03C7        ;yes.
0388-   C9 97       CMP   #$97         ;cntl-W?
038A-   D0 3C       BNE   $03C8        ;no - error
038C-   A2 28       LDX   #$28         ;yes.
038E-   C8          INY
038F-   B9 00 02    LDA   $0200,Y      ;find end of label
0392-   C9 8D       CMP   #$8D
0394-   F0 1C       BEQ   $03B2        ;fill rest with blanks
0396-   CA          DEX
0397-   D0 F5       BNE   $038E
0399-   A2 04       LDX   #$04
039B-   BD BC 03    LDA   $03BC,X      ;set up to write label
```

```
039E-   95 3B       STA  $3B,X
03A0-   CA          DEX
03A1-   D0 F8       BNE  $039B
03A3-   A0 C0       LDY  #$C0
03A5-   20 E2 FC    JSR  $FCE2      ;write it
03A8-   D0 F9       BNE  $03A3
03AA-   A9 08       LDA  #$08
03AC-   20 CF FE    JSR  $FECF
03AF-   4C 69 FF    JMP  $FF69      ;return to monitor
03B2-   A9 A0       LDA  #$A0
03B4-   99 00 02    STA  $0200,Y    ;blank fill end of label
03B7-   C8          INY
03B8-   CA          DEX
03B9-   D0 F9       BNE  $03B4
03BB-   F0 DC       BEQ  $0399
03BD-   02 02                       ;write pointers
03BF-   29 02
03C1-   20 02 FF    JSR  $FF02      ;read label
03C4-   20 8E FD    JSR  $FD8E
03C7-   AD 00 C0    LDA  $C000      ;key pressed?
03CA-   10 06       BPL  $03D2      ;no.
03CC-   AD 10 C0    LDA  $C010      ;yes - reset status bit
03CF-   4C 69 FF    JMP  $FF69      ; and return to monitor
03D2-   A2 60       LDX  #$60
03D4-   A0 70       LDY  #$70
03D6-   20 FA FC    JSR  $FCFA      ;search for label
03D9-   90 EC       BCC  $03C7
03DB-   CA          DEX
03DC-   D0 F6       BNE  $03D4
03DE-   A2 04       LDX  #$04
03E0-   BD EB 03    LDA  $03EB,X    ;set up for read
03E3-   95 3B       STA  $3B,X
03E5-   CA          DEX
03E6-   D0 F8       BNE  $03E0
03E8-   A9 04       LDA  #$04
03EA-   D0 D5       BNE  $03C1
03EC-   D0 07                       ;read pointers
03EE-   F7 07
   .
   .
   .
03F8-   4C 80 03    JMP  $0380      ;cntl-Y vector
```

©

On Interfacing An Apple II To A Heathkit H-14 ('COMPUTE', JAN/FEB/80, ISSUE2)

Mike Wiplich

I am also using an Electronic Systems serial I/O card for the Apple II. The card has been set up for 4800 baud using hardware handshake (RTS from H-14). This was done by connecting RTS from the H-14 (pin 4 on 'D' connector) to the DTR input of the serial card and using the software listed below.

A note is in order here. The software is designed to be compatible with DOS. It is installed in a separate ram between $C800 and $C85F. This ram must be supplied by the user as it is not normally part of the Apple II. The alternate solution is to relocate the code. The routine modifies the DOS output register so that all output characters are not only examined by DOS, but also by the printer driver. Upon receipt of a 'CTRL P' the printer driver will send all subsequent output to the printer until the receipt of a 'CTRL O' which will now bypass the printer. Video output is active at all times. The routine is also capable of printing upper and lower case using the 'Screen Machine' software from SOFTAPE.

To initialize the routine from machine-C800G, or from BASIC call -14336. After initializing, 'CTRL P' and 'CTRL O' work as described above.

As written the routine is set up for the serial card in slot 1 with jumper J1 connected to pin 41. The receive capability of the card is not used.

Let me add that I am quite happy with the H-14.

```
C800LL

C800-   AD 53 AA    LDA   $AA53
C803-   8D 21 C8    STA   $C821
C806-   AD 54 AA    LDA   $AA54
C809-   8D 22 C8    STA   $C822
C80C-   A9 23       LDA   #$23
C80E-   8D 53 AA    STA   $AA53
C811-   A9 C8       LDA   #$C8
C813-   8D 54 AA    STA   $AA54
C816-   A9 00       LDA   #$00
C818-   8D 07 03    STA   $0307
C81B-   A9 00       LDA   #$00
C81D-   8D 5F C8    STA   $C85F
C820-   4C F0 FD    JMP   $FDF0
C823-   C9 8F       CMP   #$8F
C825-   F0 F4       BEQ   $C81B
C827-   C9 90       CMP   #$90
C829-   D0 04       BNE   $C82F
C82B-   A9 80       LDA   #$80
C82D-   D0 EE       BNE   $C81D
C82F-   48          PHA
C830-   AD 5F C8    LDA   $C85F
C833-   D0 03       BNE   $C838
C835-   68          PLA
C836-   D0 E8       BNE   $C820
C838-   68          PLA
C839-   C9 C1       CMP   #$C1
C83B-   30 03       BMI   $C840
C83D-   4D 07 03    EOR   $0307
C840-   48          PHA
C841-   AD 91 C0    LDA   $C091
C844-   29 03       AND   #$03
C846-   C9 03       CMP   #$03
C848-   D0 F7       BNE   $C841
C84A-   68          PLA
C84B-   8D 92 C0    STA   $C092
C84E-   C9 C1       CMP   #$C1
C850-   30 CE       BMI   $C820
C852-   4D 07 03    EOR   $0307
C855-   D0 C9       BNE   $C820
C857-   00          BRK
*3
```

©

AT LAST...80 COLUMNS AND UPPER/LOWER CASE LETTERS FOR APPLE II

Michael S. Tomczyk
418 Arguello Blvd.
San Francisco, CA 94118

Where were you when the earthquake hit? I was in Sunnyvale, California when the 5.5 point Livermore earthquake rattled the headquarters of M&R Enterprises, luckily with no damage.

M&R President Marty Spergel, designer John Wilbur a..d I just grinned as we bounced back and forth, all of us thinking how ironic the quake was... since M&R has been causing some minor tremors of its own with a new intelligent board terminal for the APPLE II. The new board is called ''Sup'R'-Terminal^tm'' -- and with 80 columns and upper/lower case letters, it's definitely a super development in the personal/business computer field.

Apple owners and dealers have been speculating for months over what the new board would -- and would not -- do, so I went down for a sneak preview and here, in a nutshell, are some answers to help scotch or confirm all those rumors...

1. The new Sup'R'Terminal^tm converts the APPLE II screen to an 80 column x 24 line, upper/lower case display (5x8 dot matrix, ASCII character set)...making the APPLE II the first personal computer with this capability.
2. Installation is easy -- just plug the board into slot number 3, hook it up to a monitor and it's ready to go.
3. The board can be used with an inexpensive black & white 8MHz CRT monitor, but is not recommended for use with a normal television, which doesn't have the needed resolution or bandwidth to legibly display the smaller characters.
4. Display is in black and white only -- sorry, no color.
5. The board is fully compatible with all APPLE software (including APPLE INTEGER BASIC, APPLESOFT BASIC and PASCAL), and several other software systems (i.e. EASYWRITER^tm) are being adapted to it.
6. 2K of human-engineered, board-based editing software is included.
7. Peripherals such as disc drives and printers are fully compatible with the board, which is designed to interface with future hardware developments, such as updated disc drives, without modification.
8. When used with an APPLE II communications interface board and a program supplied in M&R's documentation manual, Sup'R'-Terminal can act as a self-contained terminal for time-sharing or other applications.
9. Effective baud rate is greater than 10,000, enabling fast clearing and scrolling.
10. Suggested retail price is $395.

The implications of the new board are enormous. Until now, APPLE users could only view 40 columns of upper case letters on their monitors, and couldn't use PASCAL (which is set up for 80 columns) to best advantage. Some owners even considered adding thousand dollar terminals to give them an 80 column, upper/lower case capability. With Sup'R'Terminal, the problem is solved.

Now wordprocessors can see the full width of their text just as it comes out on the page printer. Businessmen and accountants will be able to bring up more columns for easier tabulation and number-crunching. Even game buffs will benefit, especially if they use timesharing.

Marty Spergel sees timesharing as the most exciting application -- and it's no coincidence that M&R already manufactures a phone modem (the Pennywhistle^tm) for this purpose.

He noted that any computer owner can now access a sophisticated time-sharing system for as little as $2.75 an hour during non-primetime, but these services use a minimum 80-column format and that makes it hard to access on a 40-column monitor. He went on to explain that long programs which take up the whole screen often get scrolled off when 40-column users access it, and may be lost when the timeshare facility stops sending data. With full 80-column interface this doesn't happen, and he added that he thinks a lot more businesses will start using APPLE's for timesharing because of the new board.

SUP'R'TERMINAL NEEDS A MONITOR

The board's biggest drawback is that it has to be used with a monitor -- not a television set. Standard television screens have too low resolution and bandwidth to properly display the smaller characters which result when you go to 80 columns. This means some APPLE owners who are now using a home console will have to go out and buy a black and white monitor. With the $395 price tag for the board plus $150-200 for an inexpensive monitor, this could boost the price to as much as $600.

Recognizing that a lot of APPLE II owners will want to get a "cheap" monitor but still want good image quality, M&R built in two features which enable the user to get superior resolution from an inexpensive monitor.

System designer John Wilbur, who spent several hours demonstrating the board for me, explained that two simple screwdriver adjustments make this possible. He pointed out that under normal circumstances, a low-priced monitor displays horizontal lines much brighter than vertical lines -- for example, the horizontal line in the letter "L" shows up several times brighter than the vertical part. M&R's unique "Video Balance Circuit^{tm}" compensates by reducing the brightness of the horizontal lines in the characters. When used in conjunction with a second adjustment, which controls overall light intensity, excellent resolution is obtained.

Using both the Sup'R'Terminal and APPLE's color graphics mode requires the use of two screens -- a black and white monitor for the terminal, and a color monitor or television set for color mode. Both can be operated from the same computer, but not simultaneously. Incidentally, APPLE graphics work with the Sup'R'Terminal but only in black and white.

If the prospect of having to get a black and white monitor to use the terminal board, not being able to use your television, and maybe using two separate screens for color and black and white sounds cumbersome or expensive, you're right. But then, the major use for the terminal is business and timesharing, and in fairness, the state-of-the-art technology limitations aren't M&R's fault.

AVAILABILITY

At press time, M&R's game plan was to send 300 evaluation boards to dealers across the country so they can analyze it and show it to their customers. The boards were scheduled to go out in early February, with M&R geared up for volume production by March. Marty Spergel emphasized that deliveries on orders will be prompt.

A final question that occurred to me was whether M&R will come up with a Sup'R'Terminal for the Commodore or Atari -- but Marty just shook his head, smiled, and said, "No comment." I'm not sure what he meant by that, but you can be sure I'll be writing about it if something develops.

For more information you can write: M&R Enterprises, P.O. Box 61011, Sunnyvale, CA 94088. Next issue: Using the Sup'R'Terminal and user defined character sets... ©

Apple Software

Eric Rehnke

The last time I visited MICROPRODUCTS in Redondo Beach, proprietor Paul LaMar showed me two very interesting software packages which he sells.

The first is called the Text File Manager and is actually a two-pass disassembler which creates an assembler source from pure object code. Think about it. Remember that undocumented program you purchased and needed to modify? Or how 'bout a way to probe the mysteries of the APPLE DOS?

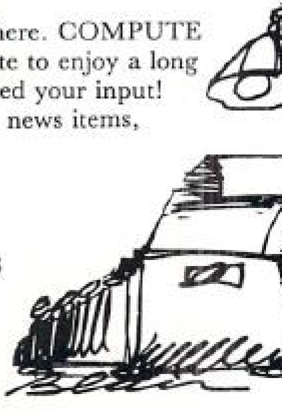
Text files that are created with the Text File Manager can then be assembled with the MICROPRODUCTS Assembler. Other features are included to make this package look even more useful. (I wish I could get this for my KIM).

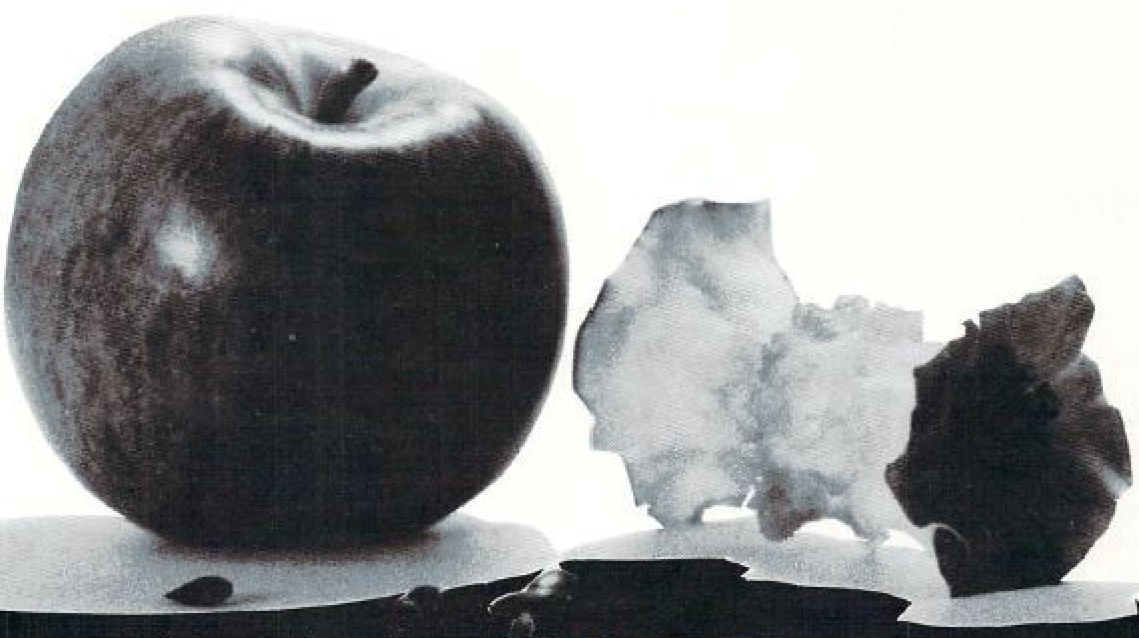
The other software package Paul demonstrated is called APPLEBUG. This is a debug tool and enhanced machine language monitor. You can single-step, trace and/or run through a program until one of four breakpoints or an 'RTS' is detected. Up to 8 memory locations may be traced during execution. Lots of other features that you can read about if you send for a catalog:

MICROPRODUCTS
2107 Artesia Blvd.
Redondo Beach, Ca 90278
(213) 374-1673 ©

Apple Authors

We know you're out there. COMPUTE wants the Apple Gazette to enjoy a long and happy life. We need your input! Send articles, reviews, news items, and club notes to:
Apple Coordinator,
COMPUTE,
Post Office Box 5119,
Greensboro, NC 27403





Introducing AppleSeed, our newest publication to whet your Apple* appetite!

We invite you to subscribe to AppleSeed - the magazine that is to the Apple II* what SoftSide is to the TRS-80. It offers the newest in software programming hints and ideas tailored especially for your computer. AppleSeed features challenging programs for both the do-it-yourselfer and the individual interested in pre-packaged programs and games . . . your own preview of the best available on the market today. A typical slice of AppleSeed consists of one major (new 16K) commercial level program (completely listed for your keying pleasure), accompanied by two or three applications for practical use or fun, supplemented by informative articles to polish your Apple*. Get right to the core of your Apple* needs and order AppleSeed today! 12 issues, 1 year, $15.00. AppleSeed is the newest member of . . .**

SoftSide™
PUBLICATIONS
6 South Street, Milford, NH 03055
(603) 673-5144

*A registered trademark of Apple Computers. **A registered trademark of Radio Shack and Tandy Corp.

THE apple® GAZETTE

APPLETIVITES AT THE WEST COAST COMPUTER FAIRE

Joe Budge
2507 Elderwood Lane
Burlington, N.C. 27215

The West Coast Computer Faire, held March 14-16 this year, was the focal point for many Apple Computer dvelopments. Several companies introduced significant new software and hardware. These new introductions should greatly enhance the Apple's capabilities. User groups from all over the world convened at the Faire to start an international user's group. The group then sponsored a full day of seminars on the Apple, with subjects ranging from the Apple's invention to its application in music and foreign language instruction.

On March 13 representatives of 60 member clubs met in San Francisco to formally initiate the International Apple Core. Directors were elected, lengthy discussions were held, and policies were worked out. Directors and officers met for the following two days arranging quite a few important details.

The International Apple Core (IAC) will be a non-profit organization dedicated to the exchange of information among Apple Computer users. This will be broadly interpreted to include technical information, software, programming information, and anything else which can benefit Apple users throughout the world. In addition the IAC will provide for communications between members and the various product manufacturers. The International Apple Core will not be affiliated with Apple Computer or any other manufacturer. Specific areas of IAC activity will include publication of the "Orchard", maintenance of a public-domain software library, support of special interest groups, support of software and hardware standards, technical support, promulgation of ethics, and organization of annual Apple - faires.

The IAC does not intend to have individual memberships. Only non-profit Apple user groups may be members. Educational institutions and other interested non-profit organizations may become associate members. They will be entitled to all the free printed information the IAC provides its members, but can not vote for directors. Commercial enterprises may become sponsors, entitled to information and participation in standards establishment. They, too, would be non-voting, but would receive preferential advertising treatment in IAC publications. Dues for members and sponsors will be $50 and $200, respectively. They will be collected every January First. Associate members pay no dues.

The "Apple Orchard" will be the IAC's official magazine. It will be published quarterly, starting September First. The "Apple Orchard" will replace future editions of "Contact." "Contact" had been sent by Apple Computer to all registered users. Unlike "Contact," the "Orchard" will be available either by subscription or by sale through clubs or stores. Apple Computer Company intends to purchase some "Orchards" from the IAC. One copy will be sent to each newly - registered Apple owner to tell them about user groups.

A number of committees and special interest groups were established to deal with specific subjects. The IAC will try to establish a hot-line for technical and software problems. Until permanent arrangements can be made, Apple owners will have to continue using current lines provided by various clubs and Apple Computer, Inc. Neil Lipson's software committee will be collecting, documenting, and distributing a diskette a month to the member clubs. These will contain public domain software and will be distributed free of charge. The IAC will also distribute to members application notes from Apple Computer and other suppliers. Special interest support includes educational and legal application groups, a handicapped usage group, and a ham radio net. Anyone wishing to contact the International Apple Core may write them at P.O. Box 976, Daly City, Ca. 94017.

The International Apple Core sponsored a series of Apple seminars during the second day of the Faire. The subjects covered a variety of fronts, from the workings of graphics and the disk operating system to applications of different languages such as Forth and Pascal. The most interest was shown in a pair of lectures by Steve Jobs and Steve Wozniak, the inventors of the Apple. They told how the Apple started as a video terminal used to play games on telecommunications networks. Being members of the Homebrew Computer Club, they became interested in microprocessors. With that, Jobs and Wozniak started to add processing and memory to the video terminal, added a hand - assembled Basic interpreter, and so on. Before they knew it, they had a microcomputer on their hands. The new machine was so popular at the meetings that Jobs and Wozniak decided to print up a circuit board. That way their friends could build the new micro too, while they were freed from supervising everyone's assembly and debugging. To recoup costs the two began selling computer boards with a set of instructions and a parts list. That worked fine for a few months. Then one store owner decided to take 50 boards, but only pre-assembled and tested. Thus was born Apple Computer Company.

Apple Computer has, obviously, grown considerably from those meagre beginnings in 1975. Upon realizing the advantage of the single-board computer over the buss machines, Apple made a few improvements to smooth out the rough edges. They added I/O decoding to the peripheral slots, a cassette interface, dynamic RAM, and better video. This created the Apple II. The Apple II has been so successful and so versatile that Apple has determined not to obsolete it. New equipment will allow Apple to easily specialize in certain areas. For example rumors abound of a new business machine similar to Radio Shack's Model II, as well as of a new Pascal teaching machine. Whether these rumors are true remains anyone's guess. Nevertheless Apple is trying to solve the problem of making a computer that is at once sophisticated and yet doesn't require high intelligence to operate.

Many fascinating new products introduced at the Faire demonstrate the Apple II's potential and versatility. The show - stopper was Microsoft's new Z-80 board. This board, which plugs into one of the Apple's peripheral slots, can supress the 6502 microprocessor and turn on its own Z-80. It uses the regular Apple memory and I/O devices. Priced at $350, the board comes complete with two diskettes containing CP/M and Advanced Microsoft Basic. Production and distribution should start in May. Fortran, Cobol, and other advanced languages should become available for the Z-80 card during the summer. On the applications side of the fence an equally significant development was Programma International's unveiling of a general accounting package. Priced around $200, it appears every bit as capable as IBM's package for the 5110. IBM's software sells at 10 times the price. Three manufacturers introduced eighty - column video boards for the Apple's text display. These are for telecommunications, word processing, and use with the Pascal editor. The boards were priced from $200 to $400, and should be available in most retail outlets at press time. With all these new developments, the next year looks to be extremely interesting for Apple users everywhere. ©

Computer House Div.

Announces

Programs for Apple

"Legal Accounting"	$1200.00
"Retail Accounts Receivable"	400.00
"Political Party Mailing List"	150.00

Programing Tools For the Commodore/Disk

"SOF-BKUP" — $40.00

Faster & Error Free Disk Copy

"FET/RECOVER" — $65.00

File Editing Tool; Examine Data Files, Fix Destroyed Pointers, Sectors may be Read, Modified, Displayed or Written — Also Files may be Re-chained

"SUPER RAM" — $20.00

Diagnostic Routine Checks Every Possible RAM Address on 8K, 16K & 32K Commodore Computers

Available on Diskette right now.....

All 3 above only $105.00 retail

ENGINEERING & MACHINE SHOP Programs for Commodore System

"Machine Part Quoting"	$280.00
"Trig & Circle Tangent"	70.00
"Bolt Circle"	25.00
"Spur Gears"	35.00
"Beams; Stress & Deflection"	145.00
"Tank Thickness" For Filament Winding	85.00

All 6 for only $495.00

And many others coming soon — including RPG for Commodore — Ask for Catalog #80-C2.

COMPUTER HOUSE DIV.
1407 Clinton Road
Jackson, Michigan 49202
Phone: (517) 782-2132

The APPLE PI Trading Library

Terry N. Taylor
Club Librarian

We have well over 3500 public domain programs on 148 disc sides in our Apple PI library. Copies of our November catalogue (showing 2780 programs) are still available for $1 plus 50 cents postage from our newsletter editor:

Rodney Nelsen
9711 Josephine St.
Thornton, CO 80229 Ph. (303) 451-7577

Later arrivals are shown in our monthly newsletters, available at $1.00 each.

While most of the approximately 800 plus programs on our Apple PI slices #1 to 20 and 178 to 200 are either original efforts by our members (both local and out-of-state) or have been modified for the Apple II from a magazine listing, there are probably a few donations that originally came from other sources. Copyrighted programs are not supposed to be in our trading library out of respect to the commercial copyright holders. So if a copyrighted version of a program appears in error, please let us know so that it can be removed. Also, since many of our members belong to several Apple user groups, if they have accidentally donated someone else's program that they did not have permission to donate to our group; and we listed it unknowingly, we are sorry again. Please inform us. The approximately 2400 programs that we have received in trade from 21 other Apple user groups are listed on our Apple PI slices #21 to 126.

Our programs were originally cataloged on a 32K Apple without the Applesoft ROM card. All of the programs on our disc slices (i.e. from 1 to 20 and 179 to 200) should have been converted to the A.S. ROM card and should now work on a 48K Apple. (In fact, some programs require 48K to operate.) To convert an Applesoft program for non-ROM use; load the program, call 3314. Then either save it or run it. To convert a program from a non-ROM card source to ROM, call 54514. This is not needed if you have either DOS 3. 2. or 3.2.1.

The Hello programs on our disc sides (i.e., Apple PI slice XXX (YYYYYYYY)) identify the type of programs on the disc (i.e. the word 'Games', 'Finance', 'Utilities', etc. will appear where the Y's are shown in brackets). The Hello program on the disk slices that we have received from other clubs is formatted the same way, except the club name or location is shown in place of the Y's when the Hello program is booted. Their name is also shown on the screen so that the donating groups receive credit for their programs.

If you have a bombout, or see possible improvements, please either note the circumstances (including the configuration of your computer) of the bombout on a card to us so that we can fix the program if needed--or improve it yourself and resubmit it as your donation. Either way, someone else will then not have to wrestle with the same bugs you did. The reason we need your configuration is that not all programs will run on every Apple (you might not have a needed accessory). As a side note, if the hires page from a previous program is still on, then (1) press CTRL-C to stop the program, (2) Poke -16298,0 to turn off the hires page, and (3) type run to start the program again. That should fix that problem. All lores Graphic programs should automatically turn off the hires page as one of their first statements. But some don't.

TO TRADE (FINALLY!!!)

FOR NON-LOCAL MEMBERS. Other Apple user groups, or individuals who just want to trade. Please either send your discs or cassettes of programs to me:

Terry N. Taylor
12319 E. Bates Circle
Aurora, Colo 80014 Ph. (303) 750-5813

I will copy your programs and return your choice of our slices on your discs and mail them right back to you. If you are a brand-new Apple user group, we will trade you two sides for one to help you get started, or even advance some programs on the understanding that you will later send some of your own programs back to us. Fair enough?

When you donate programs, please take a minute to jot down on a 3 by 5 card the following:

1) Name of the program (exactly as it will be on the disk).
2) What language the program is in.
3) What type of program it is (i.e., Music, Finance, Game, Hires, Etc.)
4) Author's name and, if different, who actually listed the program into the Apple.
5) Source of program (i.e., original program, or the Title, Page, and Issue # of the magazine copied from)
6) Length of program (if known) Also the starting address if it is a Binary program.
7) Whether the program calls up any other programs, and, if so, their names
8) What accessories, if any, are needed. (i.e., programmers aid, 48K, type of printer, disc or cassette based, etc.)
9) Finally, a short description (say 25 words or so) of what the program does.

If you don't know one of the areas, just leave it blank. Unfortunately, with over 3000 programs, not too many people can remember exactly what each program does just by looking at the title; so any help here is deeply appreciated. Still, don't let it stop you from donating your program if you feel it is a hindrance. It won't be the only one that needs some documentation.
Thanks for reading this far and happy programming. ©

Editor's Note: In future issues, we'll present interviews with Marketing personnel from Atari and Commodore.

An Interview With Taylor Pohlman Apple's Product Marketing Manager

Michael S. Tomczyk

Recently, I had an opportunity to talk with Taylor Pohlman, Apple Computer's Product Marketing Manager. Taylor is 33 years old and came to Apple less than a year ago from Hewlett Packard, where he was educational marketing manager. During our interview, he discussed two important areas -- computer dealers and consumers.

He described two basic types of dealers, noting that dealers are first of all businessmen and secondly, tend to operate in those markets they are most comfortable or experienced in.

The first type of dealer is the *hobbyist who becomes a businessman.* For these dealers, technology is the key. Since the first-round hobbyist market is pretty much saturated, Pohlman feels these dealers have to start moving from hardware -- their original orientation -- to software, and the application problems of non-hobbyist users.

The second type of dealer is the *retail businessman*. To them, selling computers is much like selling hi-fi stereo systems. A fair amount of retail audio-visual and electronics outlets are personal computer dealers, he said, and many have already developed the expertise needed to reach the computer market. A-V stores may already be dealing with schools, for example, and the educational market is a natural for them. Small stores also relate well to small businessmen because the dealers, as businessmen, have themselves encountered problems in general ledger keeping, manufacturing, accounting, etc.

"The personal computer market is emphasizing solution-oriented merchandizing as opposed to hardware-oriented merchandizing," Pohlman said, and this poses important challenges for all types of computer dealers. He said Apple has 600 to 700 computer dealers, all with different interests, sales and facilities. Some sell a single line and others a complete line, from small games to minicomputers.

"The question they all have to ask is, who are those hungry people outside the store who have problems to be solved by a personal computer? At Apple, we try to turn the dealers on to those markets -- and solutions -- whatever the dealer's 'focus'. By 'focus' I mean hobbyist, businessman, educator, home entertainment user. . . .but whatever its focus, a good Apple store doesn't confuse the customer.

"For example, the businessman doesn't want to know this microprocessor runs 20 percent faster than that one. He wants to know will it solve his problem? On the other hand, the store that just has a bunch of hardware sitting around may respond well to the hobbyist by making the hardware available on the floor and letting him look inside." (Incidentally, Taylor Pohlman's definition of a computer hobbyist is, "the individual who's interested in the man-machine interface.")

He kept emphasizing that the key to the overall personal computer market is problem-solving. In this regard, Apple's magazines and other literature are designed to trigger people's problem-solving approaches and stimulate their imagination. He also called attention to the company's seminar program, which encourages dealers to get out and give seminars where the people are -- at Rotary Clubs or real estate groups, for example. Apple provides instructional and advertising materials to dealers for this purpose.

"For the dealer to survive in the new marketplace, he's also going to have to provide service and support -- not just sales," he said. Consequently, Apple has a modular design and can be field-repaired at over 500 dealer-based repair centers, in 24 hours. If the dealer can't repair it, he can simply replace the failed part and send it to Apple for repair. This is especially important to the businessman, who can't afford to have his general ledger or other system go down for a week.

Turning his attention to consumers, he guessed that there are "tens of thousands" of Apples in homes, schools, and companies. "We're the one personal computer vendor that has achieved a truly disk-based population -- as opposed to those who are still out there hyping cassettes. Disk-based software is more sophisticated and makes the Apple more useful and powerful as a problem-solving tool.

"What I, as a consumer, want is a computer that allows me to define a problem and allows the machine to solve it in language and terms that I can understand. If the problem is solved I could care less what the machine is doing." He added that the level of computer awareness is very high but the level of computer *literacy* is not nearly high enough to create the "home computer revolution," mostly because right now using a computer means you have to:

- define a problem
- create an algorhythm to solve the problem
- write programs to express the algorhythm
- put the program in the computer
- run and debug the program

"A lot of time is wasted trying to *translate* information, and this wasted time is directly related to the reluctance -- or lack of reluctance -- of customers to get involved with a personal computer,'' he said. He went on to say that the home computer revolution will not truly arrive until you can more effectively separate the user from the intricacies of hardware and software, so he doesn't have to understand the inside of the machine or the inside of the software in order to use it.

He also spoke of a ''vast dumping ground'' where a lot of people who buy computers encounter so many stumbling blocks to using them (language, interface, hardware) that they wind up juking them, and said Apple is working hard to solve this and other problems not only through product development and dealer support, but also by encouraging such vocal forums as the International Apple Core.

He concluded by saying that he doesn't think Apple II is a mature product after two years and indicated that future Apples will maintain compatibility, as opposed to some companies which have come out with new machines which weren't compatible with earlier versions. He said, ''If people think Apple is going to somehow change it's product or it's way of doing business, they're wrong.'' ©

6502 MACRO ASSEMBLER AND TEXT EDITOR

- Versions for PET, APPLE II, SYM, KIM and ATAR I (1st quarter 1980)
- Written entirely in **machine language**
- Occupies 8K of memory starting at $2000 — Apple version with disk occupies just over 9K
- **Macro** and conditional assembly
- 36 error codes, 26 commands, 22 pseudo ops
- Labels up to 10 characters
- Auto line numbering and renumber command
- String search and string search and replace
- Copy, move, delete, load, save, and append commands

Cassette and Manual $49.95
(including U. S. postáge)

Eastern House Software

3239 Linda Dr. Winston-Salem, N. C. 27106

GIVE YOUR APPLE A BYTE OF THIS . . .

GPA ELECTRONICS' Programmable, Intelligent I/O. More than a telecommunications I/O. More than an intelligent hard copy interface. GPA's Intelligent I/O is a COMPLETE MICROCOMPUTER on a 4½"x7" card!

Completely programmable via Resident Monitor in PROM, the Intelligent I/O lets you select the operation you desire, then helps you complete the task.

FEATURES:

- ★ 6502 Microprocessor.
- ★ 2K E-PROM; 1K RAM as buffer.
- ★ Serial and parallel I/O through four ports. Device can communicate with two peripherals simultaneously; RS-232 devices can communicate with parallel devices.
- ★ Hard copy interfacing with either RS-232 or 20mA devices (Please specify.)
- ★ Accommodates all baud rates to 9600.
- ★ Selectable line feed, carriage return delay and characters per line.
- ★ Supports standard EIA modem-control signals.
- ★ Handshaking, of course.
- ★ Functions as Data Channel, SPOOLER or Programmable Filter.
- ★ Remote access to Basic.
- ★ Fully disk compatible.
- ★ Will provide secure, encrypted telecommunications with optional programming.
- ★ Complete dial-up and time sharing functions.
- ★ Allows ganging of microcomputers for multiprocessing.
- ★ Accesses and controls servos, sensors, articulated devices or transducers (with appropriate firmware, available from GPA.)
- ★ Functions as disk controller, resident assembler, E-PROM burner (with optional firmware.)
- ★ Integral power supply (5VDC @ 350mA.) Completely portable, stand-alone computer.
- ★ Fully documented, tested and guaranteed.

See your dealer today, or order direct from GPA.
Just $299.95 (110VAC) or $320.95 (250VAC)

VISA

GPA ELECTRONICS, INC.
P. O. Box 931
Laytonville, CA 95454
(707) 984-6270

master charge

* Apple is a registered trade mark of Apple Computer, Inc.

VISICALC: A Software Review

Joseph H. Budge

There are very few single programs good enough to sell computers. Visicalc is one such program. Every Visicalc user knows of someone who purchased an Apple just to be able to use Visicalc. One user wrote down the software specifications for Visicalc and took them to IBM, asking "Can you do this for me?" IBM bid at the job: Cost would be $30,000 and it would take 3 years to complete the software. Yet Visicalc is available in any computer store today for $150 and runs on $2,000 worth of hardware. More computers have been sold on its account than with any other single software product. What is Visicalc, anyway, and why is it so good?

In its simplest terms Visicalc is a numerical modelling program. The program divides computer memory into a matrix of up to 16,256 cells. Each cell can contain a string (label) or a number. So far so good; memory looks something like a multi-dimensional array in BASIC. But unlike BASIC each cell is displayed on the video monitor and can be related to any other cell by a mathematical formula. Thus cell A13 might be the sum of cells A1 through A12, while cell B13 represents the sum of the squares. Now change the value of one cell. Viola! All the other cells are instantly recomputed to reflect the change. Thus the user can set up models and interactively make changes to see the results.

The cell and formula system of memory organization is applicable to myriads of uses. Financial budgeting, with its columns and rows of numbers, fits in perfectly. Visicalc even uses 12-digit precision to allow accurate finance calculations. Loan terms, sales projections, and profit/loss statements can be successfully modelled or analyzed. But Visicalc isn't strictly a financial tool. It's capabilities are general enough to apply in many other fields. Demographers can model population trends, biologists can model biochemical systems, and physicists can model nuclear decay.

Visicalc is best known for simplifying complex business decisions. An acquaintance of mine just received a substantial raise for the forecasting work he started doing for his employer. What the employer doesn't know is that my friend takes the work home at night and solves the problems on Visicalc in 10 minutes. A large part of Visicalc's usefulness stems from its rather complex instruction set. Commands exist for inserting values and setting up formula, obviously. Other commands control screen display and split-screen displays. One part of the screen can be used to play with parameters while the other displays results. There are commands to allow cell and formulae replication as well as mass editing commands. And, naturally, models can be saved to diskette in the event that the user needs to back up a few steps.

Because of the complex instructions, Visicalc is certainly not for the faint-of-heart. All instructions consist of single or double keystrokes with minimal prompting. While this will slow down beginners, it allows experienced users to work very rapidly, Visicalc comes with excellent documentation which takes the beginner step by step through each instruction. After a week of study Visicalc operation should be second nature to almost everyone.

Personal Software has produced an excellent product in Visicalc. Unfortunately their idea of customer relations leaves a bit to be desired. Visicalc is sold on an uncopyable diskette with a 90 day warrantee. In effect one pays $150 to use a program whose performance after three months depends on luck. Three months is just about the expected useful life of a heavily used diskette. In this author's opinion the program should be good for a year or more. It should at least come with a backup so that business users don't have to suffer down-time while waiting for a replacement. Personal Software also touts their Visicalc Newsletter in their documentation. As far as I can tell no such thing exists.

There is a bug in some versions of Visicalc which users should be aware of. If you use version 1.35 or earlier to initialize a data diskette, your diskette will not initialize properly. The volume table of contents isn't set up correctly. When later data is written to the disk it will eventually overwrite the disk catalog. When this comes to pass all the data on that diskette will be lost. If you have an early version of Visicalc be sure to use Apple's regular initialization routines instead of Visicalc's.

Overall I would rate Visicalc as an excellent value for anyone with modelling to do. You will be surprised at how many applications are possible once the program is in your library. Be prepared to spend a lot of time at first studying the manual, and treat that diskette like gold. Visicalc is carried in almost every computer store, where excellent demonstrations are also available. ©

Editor's Update:

The PET/Commodore version of Visicalc is expected to be available in July. Price is unknown at this point. The Atari version is expected to be available sometime during third quarter, 1980.

Personal Software is currently reviewing their warranty policy. As Joe points out, the warranty presently provides free replacement of defective disks for 90 days. The warranty also provides for replacement of a defective disk at nominal charge ($15.00 at your local dealer, or $7.50 from Personal Software) during the first year of ownership. RCL

THE apple® GAZETTE

Dear Apple Readers:

Here's a revised and expanded Apple Gazette. We need your help and input to keep it that way. We're interested in product reviews, short programming hints and application notes, and how-to articles. Help keep the Apple Gazette healthy! Submit your articles to Robert Lock, Publisher/Editor, COMPUTE, P.O. Box 5406, Greensboro, NC, 27403. Please mark the outside of the envelope clearly with "Attn: Apple Gazette'. ©

BOOK REVIEW

Computer Graphics Primer

author: Mitchell Waite
publisher: Howard W. Sams
book # 21650/$12.95

Review by Eric Rehnke

For a really excellent introduction to computer graphics get this book. In 2 chapters the authore takes you from answering your initial questions on what graphics are and what kinds of things they can do to how they do the things they do and a survey of some of the more popular computers with built-in graphics. Lots of good data here including block diagrams of some graphic displays and information on one particular LSI color video chip, the AMI 68047.

As informative as the first two chapters are, the third chapter alone would justify the cost of the book. Lots of in-depth explanations of the Apple graphics programming interface in particular, and many programming examples. The most important stuff in the whole book was the detailed explanation of how the Apple shape tables are designed and how they can be used to build a graphic character that can be made to grow, shrink and rotate under software control. That alone should keep me busy for awhile.

Other things that were discussed in the book include other types of graphics I/0 devices like digitizer pads, light pens, plotters, joysticks, digital cameras, and image digitizers.

I can't really say enough about Computer Graphics Primer except that it would prove very useful to anyone even remotely interested in computer generated graphics, and INDISPENSABLE to anyone who owns a Apple. ©

NCC '80 And The APPLE III

Joseph H. Budge

By now everyone has heard of the Apple III, but no ones knows much about it. In May I was able to view the Apple III at NCC '80 in Anaheim. Apple computer put on a special demonstration for the International Apple Core which I was able to attend.

According to Apple, the Apple III was designed specifically for high capability use by professional and managerial people. It's features include a new Apple-designed central processor, up to 128K bytes of main memory, a self-contained floppy disk drive, and a new keyboard design with 10-key pad and programmable keys. Video allows for 80-column by 24-row text in black and white, or 40 x 24 in 16 colors as well as high resolution graphics. The III also has four analog to digital (A/D) converters, one digital to analog (D/A) converter, and a programmable character generator. Software available for the III will include Apple Business Basic, Fortran, Pascal, and Pilot. Apple Business Basic will include PRINT USING with string justification and 18-digit mathematics. Pilot is a teaching language with enhanced graphics for the III.

As a professional tool the Apple III will be promoted as part of application packages. At least three application packages are pending. The first, to be available in July, will be the Information Analyst package. It will contain an Apple III with 96K bytes of RAM and a 12-inch black and white monitor. Software will include Apple Business Basic, a mail list manager, and Visicalc III. Visicalc III is Personal Software's Visicalc with new features and functions added. Visicalc III has been additionally enhanced to utilize the Apple III's larger display and memory. Base price for the information analyst package will be $4,400.

The second application package, available in September, will be the Apple III word processor. In addition to the hardware of the Information Analyst, this package will include the buyer's choice of two printers. Either a Silentype thermal printer or a Qume-type letter quality printer will be available. The Silentype plugs directly into the back of the III, while the letter quality printer will include an interface card. Software for the package will feature a word processor and word processor training course. Base price will be $5,400 with the Silentype or $7,800 with the letter quality printer.

The third application package will be a software development package. This will come with a 128K Apple III, a 12-inch black and white monitor, and a Silentype printer. All Apple III languages will probably be included. No price or release date has been announced, but look for it in 6-9 months.

All options from each application package will be available separately. Extra disk drives and memory will probably be in most demand. The memory upgrade, from 96K to 128K, will be priced at $500. The Apple III will be available by itself, of course, although Apple has not announced the price or anticipated availability date for the hardware alone.

The III will, indeed, be a great business computer. But it's also the ultimate hobby computer. Listen to Wendell Sanders, the III's designer: "What do you expect? They gave me a blank check and told me to go build a computer. It's the ultimate homebrew!"

To understand the III, you really need to know something about it's processor. The processor is built around a 6502A running at 2MHz. The fact that there's a 6502 chip in there is a bit misdirecting. A large number of support chips are hooked onto the 6502 which lift it out of the microprocessor class and into the miniprocessor class. Output from the 6502 goes to a PIA and a ROM. Outputs from those two devices feed back to each other and to the 6502. About a dozen other logic chips are thrown in for good measure, just to confuse the issue. Apparently the ROM is the key to the whole system. It doesn't act like memory, as in the Apple II. Rather, it's a giant logic gate with many thousand possible states.

This arrangement allows for a whole host of new processor instructions and actions not possible with the regular 6502. To begin with, page zero can be located virtually anywhere in memory. An interesting possibility which results in putting banks of page zero's holding data throughout large programs. Program speed should be greatly enhanced. Of course if page zero is to be portable, the stack must be portable as well. Can you imagine what kind of Byzantine programs you could write jumping from stack to stack? Interrupts are handled differently in the III's processor as well. An added command is the equivalent of interrupt - stand-by. When this mode is enabled the processor recognizes interrupts but takes no action. Instead it flags the fact that an interrupt must be dealt with later. This allows the III to still do most everything in software but support interrupts. It's now possible, for example, to simultaneously enter text to the keyboard while disk access is going on. The processor also has more direct control over I/0 devices than has previously been possible. Scrolling is possible with machine commands. Other commands involve enabling and disabling the programmable character generator and keyboards. Finally, the most obvious benefit of the processor's enhanced logic is its ability to address 128K of memory. That's the official figure at least. Reliable sources indicate that, with some minor modification, the III is capable of handling 384K of memory in 64K chips. Whew!

Graphics freaks will love the new graphics capabilities in the III. At maximum resolution the screen has a 560 x 180 dot grid. This is exactly twice the horizontal resolution of the Apple II. Unfortunately the high resolution is only available in black and white (same for the 80 - column text). But lo-res will blow your mind! Lo-res on the III is the same resolution as hi-res on the II. But unlike hi-res on the II, the III's graphics allow for a full 16 colors, any of which can be mixed at will. This contrasts to some of the multi-color machines on the market now which only allow four colors to be on the screen at a time. And the 16 colors are available for either graphics or 40 x 24 text. Speaking of text, a real show-stopper was the programmable character generator. As you might expect, this allows the user to set up his own character set. That's lots of fun if you're into Gothic or Chinese (For some reason they didn't have a Pet graphics demo running. . .). The character generator is even more fun if you like animation. By changing the definition of characters you can easily change a character's shape. If you do this to adjacent characters in the right order you can make much larger objects move. Andy Hertzfeld showed a couple of outstanding demos of birds flying and horses running. Just like the movies. Sound from the III is literally in the 6th dimension. The speaker is run by a 6-bit d/a converter which allows true waveform generation. Unfortunately the speaker is the same crummy one found in the II. Apple ran a wire to a jack in the back so you can easily plug in an amplifier and good speaker. Apple expects to market a 12-bit A/D converter as a peripheral card. While we were discussing the sound, one of Apple's software experts noticed that the six-bit d/a converter nicely coincided with the six-bit data storage on the disk. Look for some interesting sound disks to come along soon.

The disk drive itself is pretty much the same as the Disk II we're already used to. To please the FCC's radio emission standards a change has been made that squelches direct compatability with Disk II drives. To keep radio emission down you can't have a million cables running out the back of the computer. So the disk interface was changed such that disks daisy-chain. That way only one cable comes out of the computer, for the second drive. The third plugs into the back of the second, and so on down the line. After hearing this several spectators suggested that the first non-Apple product for the III will be conversion kits for Disk II's. Data on diskettes will be compatible between the language system II and the III.

Apple received some criticism for introducing the III with only 143K byte drives. They apparently feel that this won't be a problem in early applications. Even more tantalizing is their elaboration of this point: "We will also expand our line of mass storage peripherals in the future, since some applications need several megabyte of memory."

No system would be complete without some software to make it work. This is especially important in the Apple III, where all kinds of strange tricks are going on inside the processor. In anticipation of this problem Apple provides the Sophisticated Operating System (SOS) for the III. SOS manages memory, peripherals, the keyboard and screen, graphics drawing, and interrupt handling. Most utility functions are built into it. Being configurable, SOS only pulls in from disk those parts of itself needed for the tasks at hand. With most programs SOS will take up about 14K of memory. One of its main functions is to efficiently manage the memory that's left. SOS automatically finds and allocates free memory for program use. If a memory sensitive function, such as graphics, is switched on, the SOS will reallocate memory around the graphics page. SOS's other main function is to handle I/O. It keeps track of what peripheral card is in what slot and has built in programs to control the cards. Peripherals can be interrupt driven for maximum efficency, SOS also has built in routines for graphics and sound. SOS thus provides a foundation for higher level languages. The user's application software can be truthfully independent of the hardware.

Most Apple II owners will be interested in compatability between their computers and the III. For this reason the III comes with an Apple II emulator mode. The emulator mode copies the II as faithfully as possible. It allows you 48K of RAM with either Applesoft or Integer Basic available in "ROM". Graphics and text appear as on the II; 6 color hi-res, lores, and uppercase only text. The microprocessor clock even slows down to 1MHz. Put in the emulator disk and it effectively performs a frontal lobotomy on the III.

Naturally there are some differences between the emulator and the real thing due to hardware differences. There is no cassette recorder port, for instance. Game paddles will need adapter sockets, and there are no annunciators. There are no slots 0, 5, 6, or 7. The emulator supports two disk drives which appear from the software's viewpoint to be in slot 6,1 and 6,2. The physical dimensions of peripheral cards in the III are different: higher and shorter. Your Mountain Hardware clock and Sup'R'Term won't fit in the III - but you won't need them anyway. Most other peripheral cards for the II should work in the III since most of the buss lines are identical. The III uses the no-connects, DMA, and User1 lines for its own purposes.

Apple has made a full-scale commitment to continued support of the Apple II. Taylor Pohlman, the product manager, told us that Apple intends to sell as many Apple II's in the next 12 months as have already been sold to date. Two of the new languages, Fortran and Pilot, will be available for the II before they are available for the III.

The Microsoft Z-80 SoftCard. Leading a Whole New Lineup for Your Apple II.

Plug the new Microsoft Z-80 SoftCard into your Apple II™ and start using all of the system and application software written for Z-80 based computers. Software that you could never use before on your Apple II.

The SoftCard actually contains a Z-80 processor and lets you switch between the Apple's 6502 and the Z-80 with simple commands, so you can use software written for either processor.

Starting with Two Software Standards. Versatile CP/M,® the most widely used microcomputer operating system ever, is included on diskette in the SoftCard package, ready to run on your Apple II.

You get Microsoft's 5.0 BASIC too, the most powerful version to date of our famous BASIC interpreter.

PRINT USING, 16-digit precision, CALL, and CHAIN and COMMON are just some of the major BASIC features you'll add. Applesoft's graphics extensions are still included.

More Power Down the Line. You can get even more programming power and versatility by adding Microsoft's FORTRAN, COBOL, BASIC Compiler and Assembly Language Development System. All are available separately to run with the SoftCard system.

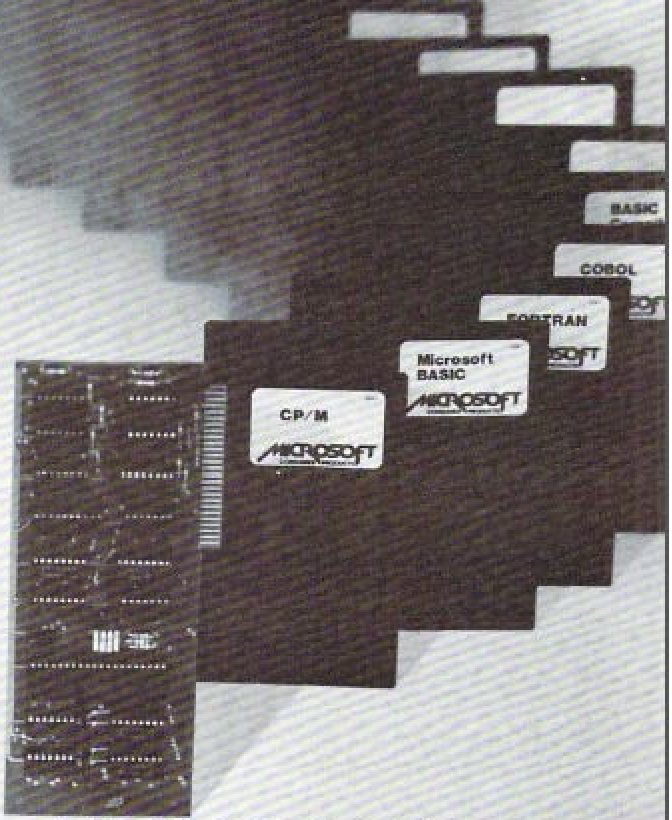
And the whole host of CP/M-based business, scientific and educational applications can be easily transferred to your Apple with SoftCard.

The Microsoft Z-80 SoftCard is compatible with most every Apple product from the Apple II to the Apple II Plus, Language Card and peripherals. Independent peripherals for the Apple are supported as well. The SoftCard package requires a system with 48K and a disk drive.

Line up a SoftCard demonstration at your Microsoft Consumer Products dealer today. They'll be glad to show you how the Z-80 SoftCard and your Apple computer combine to form a system that can't be beat for either practicality or pure pleasure by any personal computer available today. Or give us a call, 206/454-1315, for more information.

But act quickly. At the low price of $349 for SoftCard, CP/M, Microsoft BASIC and complete documentation, you may have to stand in line to get one!

™Apple II is a trademark of Apple Computer, Inc.
®CP/M is a registered trademark of Digital Research.



MICROSOFT
CONSUMER PRODUCTS
10800 Northeast Eighth, Suite 507
Bellevue, WA 98004
(206) 454-1315

And Apple intends to release sometime this summer a Pascal interpreter that will run in a 48K Apple II with no Language System. Thus all Apple owners will be able to run compiled P-code. Apple sees the II as a beginner's system and as a basic workhorse for small applications.

Apple's introduction of their own new computer was done in the Grand Style. NCC is a very political convention, where pecking orders and pull count heavily for floor space and location. Both are important, since in excess of 80,000 people attend the convention. As you might expect, NCC wanted to put Apple in the personal computing section. Apple, on the other hand, wanted to insist that the III was a small business comupter. The haggling went on long enough that Apple finally wound up as neither; they were put in the "overflow" section in the basement garage of the Disneyland Hotel. The garage was over a mile from the main convention floors. Apple wasn't even listed in the convention's guide to exhibitors. Nevertheless the Apple booth was the second - most popular booth at the show; only IBM drew a bigger crowd. At Apple's booth one had to stand in line 20 minutes just to get near a machine! By the show's third day officials had grown so tired of "Where's Apple?" questions that 10 signs were up on the main convention floor giving instructions on how to get over to Apple. So much for politics.

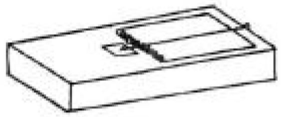
Several interesting products were introduced for the Apple II by other manufacturers at NCC '80. Mountain Hardware announced their new music board. Actually, it's two boards and a light pen. The boards allow 16 voice music synthesis, with each voice having a fully programmable waveform and envelope. Software included allows input of sheet music in standard notation using the Apple's graphics screen and the light pen. Output is in stereo, naturally, and sounds excellent. The complete system will be priced at $545. Mountain Hardware also had a bit of information on their Apple buss expansion module. Contrary to ads you may have read, the expansion box won't be ready for at least 4 months. It will have its own power supply, a switching supply slightly greater than Apple's (my sources indicate that this supply is manufactured in the Far East and is difficult to get in large quantities). The box will have 9 slots, one to replace the slot which the box plugs into and eight more (0'-7'). It employs the User1 line to disable the boards on the Apple itself when appropriate. PR#14!

Programma showed the prototype of their dual sided 8" disk controller. It's a neat board, absolutely crammed with chips. To overcome power supply limitations, the controller card gets all its power from the disk drives. That should be ready in about 6 months. Programma also showed their new text editor, PIE 2.0. It does almost everything you can imagine (I didn't find out if it works with LISA). One of it's nice features is footnoting. Unfortunately, for some reason it's still a line-oriented editor. Oh well, it's probably the best line-oriented editor you'll ever see.

Stoneware Computer Products was demonstrating the prototype of their Data Base Management System. It is indexed so that any data can be found in 6 seconds. Each entry can be up to 1K long, and have up to 100 fields. Data types can be defined, and the user has complete control over report generation. The system has been elegantly human engineered to be useable by the non-programmer. A first version will be released this summer which will support one disk full of data. The second version, which will soon follow, will support up to 25 megabytes and have many other features.

Epson America introduced a new printer for the hobbyist and small business market. Called the TX-80, it's an 80-column dot-matrix impact printer with either tractor or friction feeds. The printer comes with a Centronics - type parallel port and options for RS-232 serial or IEEE 488 interfaces. It prints at 125cps. Epson designed the printer with high reliability and ease of use in mind. The ribbon comes on standard typewriter spools, for instance. Of the thousands of TX-80's sold in Europe, less than 1% needed service of any kind. Of course the added quality costs: the printer is expected to sell in the $750 range when marketing begins in this country. ©

TRAP 65



For PET, APPLE, SYM, OSI, etc.
Any 1 mhz 6502!

— Prevents from "hanging up" via execution of **unimplemented opcodes!** Causes your monitor to **display location** of bad op code!
— Our computers have stopped crashing!
— **Debugging easier!**
— **Easy to install** — plugs into 6502's socket.

TRAP 65 is currently being used to insure that there are no bad opcodes in programs before submittal for ROM masks!

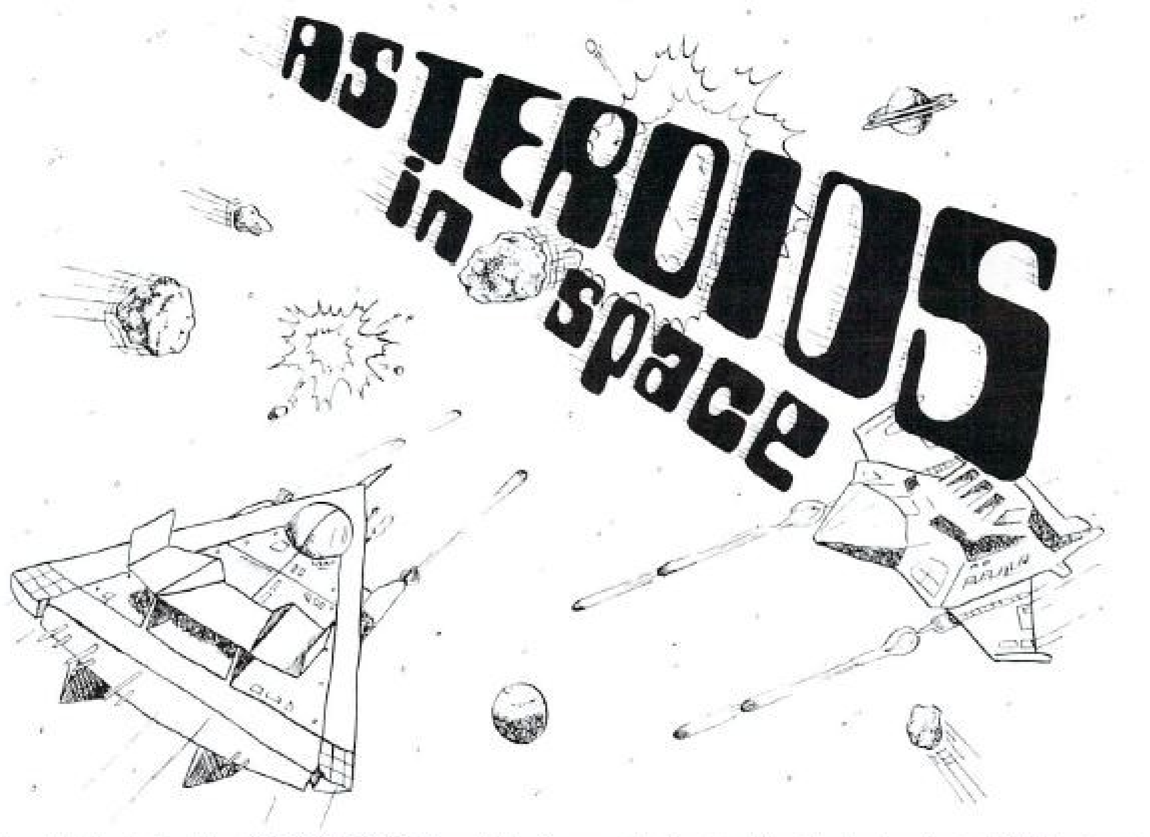
We are using TRAP 65 in experiments on extending the 6502's instruction set — PHX, PHY, INCD.

$149.95 (Add $4.00 postage, $10.00 foreign)
Phone [919] 748-8446 or 924-2889

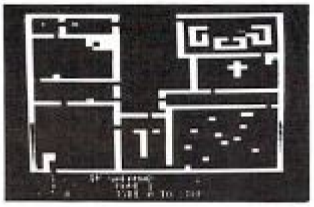
Eastern House Software

3239 Linda Dr. Winston-Salem, N. C. 27106

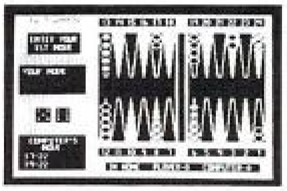
Exciting, entertaining software for the Apple II and Apple II Plus*



If you liked "Invaders", you'll love ASTEROIDS IN SPACE by Bruce Wallace. Your space ship is traveling in the middle of a shower of asteroids. Blast the asteroids with lasers, but beware — big asteroids fragment into small asteroids! The Apple game paddles allow you to rotate your space ship, fire its laser gun, and give it thrust to propel it through endless space. From time to time you will encounter an alien space ship whose mission is to destroy you, so you'd better destroy it first! High resolution graphics and sound effects add to the arcade-like excitement that this program generates. Runs on any Apple II with at least 32K and one disk drive.
On diskette — **$19.95**



FRACAS™ by Stuart Smith. A fantastic adventure game like no other — up to eight players can participate in FRACAS at the same time. Journey in the land of FAROPH, searching for hidden treasure while warding off all sorts of unfriendly and dangerous creatures like the Ten Foot Spider and the Headless Horseman. You and your friends can compete with each other or you can join forces and gang up on the monsters. Your location is presented graphically and sound effects enliven the battles. Save your adventure on diskette or cassette and continue it at some other time. Requires at least 32K of RAM. Cassette: **$19.95** Diskette: **$24.95**



BATTLESHIP COMMANDER™ by Erik Kilk and Matthew Jew. A game of strategy. You and the computer each start out by positioning five ships of different sizes on a ten by ten grid. Then the shooting starts. Place your volleys skillfully — a combination of logic and luck are required to beat the computer. Cartoons show the ships sinking and announce the winner. Sound effects and flashing lights also add to the enjoyment of the game. Requires at least 32K of RAM. Cassette: **$14.95** Diskette: **$24.95**

FASTGAMMON™ by Bob Christiansen. Sound, hi res color, and cartoons have helped maked this the most popular backgammon-playing game for the Apple II. But don't let these entertaining features fool you — FASTGAMMON plays serious backgammon. Requires at least 24K of RAM. Cassette: **$19.95** Diskette: **$24.95**

QS **QUALITY SOFTWARE**

6660 Reseda Blvd., Suite 105, Reseda, CA 91335

WHERE TO GET IT: Call us at (213) 344-6599 for the name of the Quality Software dealer nearest you. If necessary, you may order directly from us. MasterCharge and Visa cardholders may place orders by telephone and we will deduct $1 from orders over $19 to compensate for phone charges. Or mail your order to the address above. California residents add 6% sales tax. SHIPPING CHARGES: Within North America orders must include $1.50 for first class shipping and handling. Outside North America the charge for airmail shipping and handling is $5.00 — payable in U.S. currency.

*"Apple II" and "Apple II Plus" are trademarks of Apple Computer, Inc.

APPLE WRITER™

A Review Of The Text Editing System From Apple Computer, Inc.

David D. Thornburg
Innovision
P.O. Box 1317
Los Altos, CA 94022

The use of personal computers in small businesses is clearly bringing tremendous sophistication to tasks which a few short years ago were being carried out manually. To take just one example from the data processing (DP) world, the Apple II is being used along with Personal Software's Visicalc program for numerous data processing applications in thousands of businesses.

In addition to data processing, all businesses (and homes for that matter) generate documents which have to be sent to others in the form of letters, reports, or memoranda. The creation of these documents is aided by ''Word Processing'' (WP) software. It is reasonable to expect the computer that handles the data processing (DP) tasks to handle the WP tasks as well, since the primary ingredient for doing the job is software. There are some hardware constraints, as we will see, which can make a good DP computer less than perfect for WP applications, but it is still the software which carries 99% of the load.

Since Innovision uses a 48K Apple II for its business applications, I was only too happy to get my hands on the Apple text editing system, ''Apple Writer'', when it was released several months ago. This text editing system comes with two disks (one for back-up) and a manual which continues the fine tradition of Apple in that the documentation is superb. Since this package was priced inexpensively at $75, I did not expect to have all the features one expects from a $16,000 dedicated word processor. What I did find was a compact text editing system with a lot of the essential features needed for the rapid preparation of letters and small (under 100 page) reports.

HARDWARE CONSTRAINTS PROVE CHALLENGING. . .

If you are an Apple owner, you may have run up against two Apple characteristics which might make you skeptical of this machine's ability to be used for document creation. These characteristics are the lack of lower case character entry from the keyboard, and the inability of an unmodified Apple II to display lower case characters on the screen. These constraints, coupled with the limitations of a forty character by twenty-five line display, might make one shudder to think of doing text editing on such a machine. My experience suggests that things aren't as bad as they might first appear - and I use this system for text editing every day of the week.

Apple uses its own convention for character presentation on the screen. Upper case letters are shown in reverse field (black on white) and lower case letters are shown as normal (white on black) upper case letters..The keyboard is handled in the following manner. All alphabet keys are accepted as lower case letters. Upper case letters are obtained by preceding a letter with the ESC key. Since the ESC key is located in the same column as the SHIFT key, this doesn't cause too much trouble. Normally shifted characters (the punctuation marks above the numerals, for example) are obtained with the SHIFT key. As characters are entered into the computer, they are displayed on the screen just as they are typed in. This means that words will be broken at line boundaries (40th character position). Thus making it hard to do extensive proof reading from the screen. Since the goal is to produce printed documents, this is not a terrible inconvenience, as I am willing to perform proofreading tasks on the printed output.

Please note that while I find the constraints imposed by the Apple hardware tolerable for this application, I am not suggesting that they could not be improved. In fact, I think that all personal computers should have full upper and lower case keyboards and displays. What I am saying is that, given the constraints imposed by the hardware, the Apple Writer word processing system is easy to learn and use.

BRINGING THE SYSTEM UP . . .

If you have Applesoft and an auto-start ROM, the Apple Writer is virtually crash-proof. Accidental pressing of the RESET key, for example, does nothing more serious than moving you to the menu. Note, however, that this is not true unless you have the auto-start ROM and Applesoft BASIC, and that it is a bad practice to return to the menu this way. If the system is brought up with the disk in place, the user is automatically presented with a menu of commands. The commands allow one to create a new document, bring in an existing document from the disk, save a document on the disk, and bring up the printer program (which will be discussed later). The ''Edit'' command clears the menu window and places the document to be edited on the screen.

Since it is important to be able to move the cursor around on the screen to edit various portions of a document as well as to bring new portions of the document into view, text editing systems need some method for cursor control. Apple Writer lets the user move between text entry and cursor control modes by use of the ESC key. If the ESC key is pressed once, the cursor becomes a carat ( ). This symbol indicates that the next typed letter will be printed in upper case. If the ESC key is pressed twice in succession, the cursor becomes a plus sign (+) which indicates that the system is now in the cursor control mode. The cursor can be moved anywhere on the screen using the I, J, K, and M keys to move the cursor up, left, right, or down respectively. Once the cursor is in the desired place, the text entry mode can be entered by pressing any of the other alphanumeric keys. A little practice makes this seem quite natural.

Large scale cursor motion is available from several control functions. CTRL B, for example, moves the cursor to the beginning of the document.

The deletion of unwanted text can be carried out on a character-by-character basis, as well as by word, or by paragraph. These capabilities are easy to master. As characters are deleted they are stored in a 255 character buffer so that they can be re-entered later if desired. The easy way to move a word, for example, is to delete it from its old position with the back arrow key (<-) and to then move the cursor to the location at which the word is to be entered. Once this has been done, the word may be entered with the forward arrow (->) key.

More sophisticated editing tasks, such as search and replace, file insertion, and others, are also implemented in Apple Writer.

In addition to the documentation provided in the 70 page manual accompanying the software, Apple also provides an on-line tutorial to bring you up to speed with "hands-on" practice. All in all, this is a fine text editor for simple document creation.

Since the keystrokes are only saved on the disk as a result of your formal request to that effect, I find it useful to save my documents every fifteen minutes or so, just in case the power goes out. The user can find the amount of space remaining in free memory by typing CTRL F. With a 48 K Apple, one should be able to create about sixteen pages of text in one file.

GETTING IT PRINTED . . .

Sooner or later, you will want to take the jumble of broken words which appear on the screen and turn them into a nicely formatted document. During the document creation process you can enter non-printing commands (which do appear on the display) to let the printer do some fancy stuff like centering, fill justification, etc. These commands occupy one line each and are preceded by an exclamation point. For example,

```
!cj
```

is a command to tell the printer to center all the text which follows. Since each command stays active

ACCOUNTING SOFTWARE FOR YOU

GENERAL LEDGER

If you are a business person who is looking for ultimate performance, take a look at this outstanding General Ledger package from Small Business Computer Systems.

Our package features six digit account numbers, plus thirty-one character account names. We have ten levels of subtotals, giving you a more detailed income statement and balance sheet with up to nine departments. Either cash or accrual accounting methods may be used. The cash journal allows a thirty-three character transaction description and automatically calculates the proper off-setting entry. You may print the balance sheet and income statement for the current month, quarter, or any of the previous three quarters. Also, this year's or last year's total are included on the income statement, depending on the current month.

There is virtually no limit on entries, since you may process them as often as you like. Two thousand (1,000 from G/L. 1,000 from any external source) can be processed in one session.

ACCOUNTS RECEIVABLE

Sound business management requires you to keep up-to-date reports regarding the status of your accounts receivable.

Now, from the same company that revolutionized accounting on the Apple II computer, with their conversion of the Osborne/McGraw-Hill General Ledger program, you may now obtain the Accounts Receivable package you have been waiting for.

Our package allows you to assign your own alphanumeric customer code up to six characters. Date of the last activity, as well as amounts billed this year and last year are maintained. This Accounts Receivable system maintains six digit invoice numbers, six digit job numbers, invoice amount, shipping charges, sales tax (automatically calculated), total payments as well as progress billing information. You may enter an invoice at any time; before it's ready for billing, after you have billed it, and even after it's paid. This package also prints reports which list the invoices you have not billed yet, open items, paid items, and an aging analysis of open items.

In the final analysis, making your bookkeeping easier is what our software is all about. With our General Ledger package you can format your own balance sheet and income statement. Department financial statements may be formated differently. You have complete freedom to place titles and headings where you want them, skip lines or pages between accounts and generate subtotals and totals throughout the reports — up to ten levels if you need them. Accounts Receivable is designed to provide you with complete up-to-date information. The program will print customer statements as well as post invoice amounts to any of the accounts maintained by our General Ledger package. These packages will support any printer/interface combination. General Ledger requires one hundred ten columns, Accounts Receivable requires one hundred thirty columns.

Suggested Retail:
Individually $180.00
Together $350.00

SMALL BUSINESS COMPUTER SYSTEMS
4140 Greenwood
Lincoln, Nebraska 68504
(402) 467-1878

Available from your local Apple Dealer or contact SBCS until superceded by another one, it is easy to enter text in the midst of a long document without worrying about how it will be formatted.

Once the document is to be printed, one enters the print mode from the editor menu. At this point, one gets a print menu which includes the functions Print and Continue. If the Print function is selected, a list of default printer settings is displayed on the screen (left and right margins, lines per page, interline spacing, etc.). The user has the option of changing any of these parameters, in which case the parameter set is stored on the disk and becomes the new default. This feature is quite valuable, since there are likely to be few changes in overall document format once your printer is set up the way you like it.

Once the printer parameters are set, the user is given the option of entering a page heading which appears on each page of the document. Finally, the printer springs to life and the document gets put on paper for all to see.

Large documents which require more than one file to hold them can be printed with no trouble at all by using the Continue command from the Print program menu. A file can end in mid sentence, in which case the remainder of the text can be loaded into the computer and the printing resumed with no problem whatsoever. The only problem with printing large documents is that the process requires lots of operator interaction. This becomes especially noticeable if several copies of a multiple file document are being printed.

We have used Apple Writer with a Qume daisy wheel printer and with the Comprint electrosensitive printer connected to the Apple parallel interface card. In both cases the documents turn out beautifully, although I think there is a small bug in Apple Writer which makes it a little tricky to use the Comprint printer in the single page mode.

As you may know, the Comprint printer uses roll paper. If you want to print documents one page at a time (so that you can easily tear them off at 11 inch lengths), you must use the "single page" mode during printing. When this is selected, there is an extra line feed command sent just prior to printing the first page. Unless the user anticipates this and puts the printer in the "no-print" mode just prior to printing the first page, the resulting document will be improperly spaced.

Given that this product has only recently been placed on the market, I am happy that this is the only "bug" I have uncovered to date.

AND IN CLOSING

I have looked at other text editors for the Apple, some of which were overloaded with features. Given the hardware limitations of the Apple II, I feel that Apple Writer is a very useful document creation tool.

After all, how do you think this manuscript was created? ©

GIVE YOUR APPLE A BYTE OF THIS . . .

GPA ELECTRONICS' Programmable, Intelligent I/O. More than a telecommunications I/O. More than an intelligent hard copy interface. GPA's Intelligent I/O is a COMPLETE MICROCOMPUTER on a 4½"x7" card!

Completely programmable via Resident Monitor in PROM, the Intelligent I/O lets you select the operation you desire, then helps you complete the task.

FEATURES:

- ★ 6502 Microprocessor.
- ★ 2K E-PROM; 1K RAM as buffer.
- ★ Serial and parallel I/O through four ports. Device can communicate with two peripherals simultaneously; RS-232 devices can communicate with parallel devices.
- ★ Hard copy interfacing with either RS-232 or 20mA devices (Please specify.)
- ★ Accommodates all baud rates to 9600.
- ★ Selectable line feed, carriage return delay and characters per line.
- ★ Supports standard EIA modem-control signals.
- ★ Handshaking, of course.
- ★ Functions as Data Channel, SPOOLER or Programmable Filter.
- ★ Remote access to Basic.
- ★ Fully disk compatible.
- ★ Will provide secure, encrypted telecommunications with optional programming.
- ★ Complete dial-up and time sharing functions.
- ★ Allows ganging of microcomputers for multi-processing.
- ★ Accesses and controls servos, sensors, articulated devices or transducers (with appropriate firmware, available from GPA.)
- ★ Functions as disk controller, resident assembler, E-PROM burner (with optional firmware.)
- ★ Integral power supply (5VDC @ 350mA.) Completely portable, stand-alone computer.
- ★ Fully documented, tested and guaranteed.

See your dealer today, or order direct from GPA.
Just $299.95 (110VAC) or $320.95 (250VAC)

VISA

GPA ELECTRONICS, INC.
P. O. Box 931
Laytonville, CA 95454
(707) 984-6270

master charge

* Apple is a registered trade mark of Apple Computer, Inc.

Apple II Rom Card Documentation

Jeff Schmoyer
Andromeda
Computer
Systems

The only schematic that has not been made available by Apple (that I know of) is for their ROM card. The reason for the presentation at this time is not out of looking for something to do, but to obtain the solution to my blowing out the card for the third time. In each case the computer acted the same, coming up with the screen full of junk in the Lores graphics mode with RESET having no effect. Each time removing the ROM card alleviated the problem.

Also in every case the perpetrator was found to be the 74LS09 that provides the inhibiting of the main board ROMs and the output enable for the data buffer on the card. Apparently this chip inhibits the main board ROMs while not enabling its own ROMs. The processor can't find anything to execute so it hangs up. Replacement of this chip restored the system to its normal operation every time.

While we're here I might as well explain how the board works and about the two options present on it.

The 74LS74 flip-flop on the card determines whether the on-board or off-board ROMs will be used. When RESET comes through the system, the flip-flop is either preset or cleared depending on the position of the switch on the back. The other way of activating the flip-flop is through the use of Device Select. The low order address bit, A0, determines whether to set or reset it in conjuction with a C08X access.

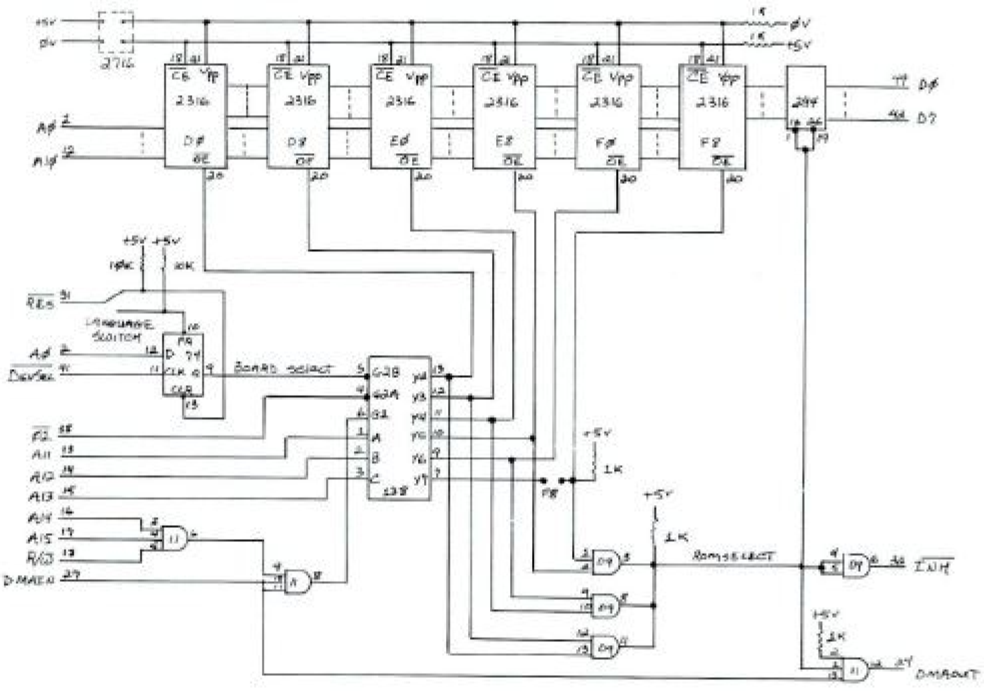
The 74LS138 IC decodes the high order address lines, with enables also coming from 01 and R/W, to provide output enable to the proper ROM. If any of the ROMs are enabled, the 74LS09 activates the INH line to inhibit the main board ROMs, and it enables the 74LS244 output data buffers.

Another service the card performs is the daisy chaining of the DMA line through it.

On to the option jumpers. One of the jumpers on the board permits the use of a ROM in the F8 position. This jumper is labeled F8 and is located near the 74LS138. If a blob of solder is melted across it, this position will be used whenever the card is selected. This could contain an Auto-start ROM if a standard monitor is on the main board, or it could contain the old monitor for an Apple II Plus.

The other two jumpers, which are located in the upper right corner of the card, select whether you want to use 2716 (programmable ROMs) or the usual 2316 (Apple ROMs) chips on the card. If both of them are jumpered, 2716's may be used. Both kinds of chips can not be used at the same time. Please notice that on the schematic, 2716 notation is used for the ROMs.

One final note. On at least one of the occasions the 74LS09 blew out, nothing was done to the computer other than picking the whole thing up and moving it as a unit. So don't assume someone pulled out a board with the power on. (I did that the last time!)



©

Review: DDT (DISC DRIVE TIMER)

Morton Technologies, Inc. $19.95 P.O. Box 11129 Santa Rosa, CA 95406

Reviewed By:
Terry N. Taylor, Apple Pi, Denver, CO

Most of us have seen Disco-Tech's ads on their drive timer for the Apple, and most of us have asked why we should purchase this product when a perfectly good public-domain program called DSPEED is already available. The answer is simple: The program is good and the documentation is excellent-- a 24-page booklet that covers the following topics:

1. Introduction
2. Running the Apple DDT Program
3. Adjusting Disc Drive Motor Speed
4. Care of Magnetic Media

This documentation alone justifies the price of $19.95 for the package. There have been several articles in the Apple user group newsletters on how to adjust your drive's speed with only two screwdrivers and DSPEED, but they are not generally available. In addition, this booklet does a far better job than do the articles.

OK, so the documentation is better; how about the program? It's better too. I have lost count of the number of times that I have made additional copies of DSPEED for those who have forgotten to put a blank diskette in the disc drive and thus ruined the program. You don't need to worry about changing diskettes with DDT. This program offers two major options. First, it allows the user to analyze the motor speed of the disc drive in question. It displays both the motor speed in revolutions per minute and the percentage of error (from 300 rpm, which they say is the correct speed) for the disc drive. DDT averages the disc speed over 40 resolutions, taking about 12 seconds to present the analysis.

The second major option updates the motor speed about once per second and graphically displays the results. Since the program asks you for an rpm range, you are able to adjust the disc motor speed very accurately, from a large-scale spread down to a very small dispersion, which watching your video screen.

To adjust my three disc drives, with either program, took less than 20 minutes using only two screwdrivers (one of which should be a Phillips), so this is not a complicated procedure; and, better yet, does not void your warranty if you follow the very good directions in the DDT booklet. I found only one minor fault with the instructions: if you have an early disc drive (for example, one with a serial number of 1800 as contrasted to one with a serial number of 6000-7000), then the trimpot screw does not face to the side, but faces down. You have to remove four more Phillips screws in order to reach the adjustment screw. Then you must place the disc drive so that the trimpot screw overhangs the edge of the table. Now you can adjust the disc drive's speed while the drive is in a normal position.

Since I was curious not only as to how this program compared to DSPEED but also to the disc speed adjustment program on Apple's diagnostic test disc, I went to a local computer store where we compared all three programs on two different disc drives. The results are shown below. While each one is different, they all result in the same conclusion: Each drive is slightly slow; but, since they are well within tolerances, the two drives did not need adjusting.

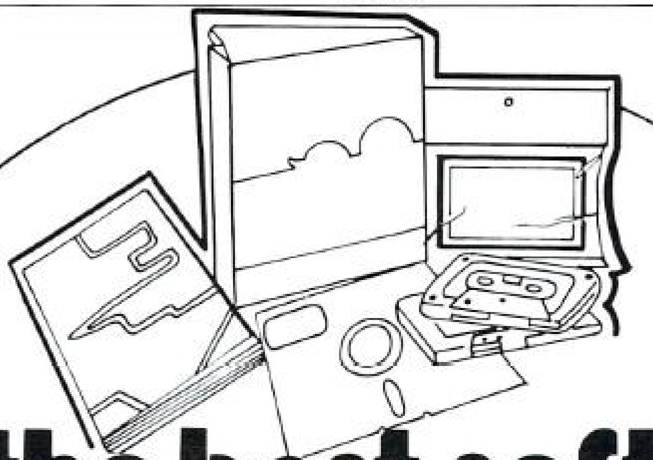
Program	Disc Drive 1	Disc Drive 2
Apple's	0 to -3	0 to -2
DSPEED	-1 to -4	0 to -2
DDT	(299.1)	(299.5)

Why should you adjust your drives? If you only have one disc drive, it's not too important to adjust your disc speed unless it's way out of whack. It is important when you are using two disc drives to copy programs from one to the other. If you are using the fast copy method (about 28 seconds) with two disc controller cards, then it is critical that the two drives be closely aligned. If you want all of the data to be properly transferred, the drives should be within one percent of each other, which is less than three revolutions per minute difference. Disco-Tech states that an incorrect motor speed can cause data to be lost when the program is either loaded or saved to disc. Unfortunately, the disc motor is easily jarred out of alignment, so this program should be run every time you transport your disc drives. Otherwise, a monthly check is probably sufficient.

When you order DDT, be sure to specify either Apple or TRS-80 as the price is the same for either ($19.95). The TRS-80 version is also available on cassette for $14.95; the Apple version only comes on diskette. Disco-Tech is a division of:

Morton Technologies, Inc.
P.O. Box 11129
Santa Rosa, CA 95406

If you are a resident of California, you must include 6% sales tax. Happy computing. ©



Buy the best software and get the fastest service - from us.

Apple Software

GHOST TOWN ADVENTURE
Scott Adams' newest! Thirteen treasures hidden in a ghost town. Be ready for a lot of surprises and real ghosts.
Cassette/24K/M $14.95

SUPER INVASION
The best invaders game available on home computers! Sensational Software.
Cassette/32K/M $19.95

WINDFALL
Put yourself in charge of Engulf Oil. Try to keep consumer prices down and still maintain a profit. Edu-Ware.
Cassette/32K/A $14.95
Disk/32K/A $19.95

VISICALC
Calculate sales projections, income taxes, personal budget, cost estimates, even balance your checkbook without a pencil and paper! Personal Software.
Disk/32K/A $150.00

ANDROID NIM
Wonderfully animated version of the popular game called Nim.
Disk/24K/M $17.95

MAGIC PAINTBRUSH
Incredible graphics are easy with this program. Use your game paddles to indicate two points on the screen, press a button to draw a line between them. Use fill mode to fill in areas of the screen with color, paintbrush mode (nine different size brushes) to paint on the screen. Save your drawings or shape tables as B files. Includes Slot Machine and Applesoft Invaders to show what you can do. MP Software.
Disk/32K/A ROM $29.95

SARGON II
The chess program that beat all the others. Faster response time and even a hint mode! Hayden.
Cassette/24K/M $29.95
Disk/48K/M $34.95
Also available Sargon I
Cassette/24K/M $19.95

THREE MILE ISLAND
Can you prevent the radiation from leaking into the air while still making a profit? Only you can tell in this nuclear holocaust simulation. MUSE.
Disk/48K/I $39.95

BEST OF MUSE
Tank Attack, Escape, Maze Game, Music Box, and Side Shows, all on one disk.
Disk/16K/I $39.95

MICROSOFT ADVENTURE
The people who wrote BASIC for all the personal computers, now bring you a version of the original Adventure. You no longer need a PDP-10 for the power of the original game!
Disk/32K/M $29.95

BASEBALL
High resolution sports excitement as you control players. Balls and strikes tallied as you control pitches and swings. MUSE.
Cassette/16K/A-ROM ... $14.95

Pet Software

RESCUE AT RIGEL
Search the moon base and rescue Delilah Rookh from the High Tollah. (Automated Simulations)
24K Pet Tape $19.95

DATESTONES OF RYN
Cross swords with a band of dastardly robbers. Real time adventure with graphics. (Automated Simulations)
16K Pet Tape $14.95

STARFLEET ORION
Command a starfleet! 2 player game system includes rule book, battle manual, control sheets, 2 programs, 22 space ship types and 12 play tested scenarios.
8K Pet Tape $19.95

INVASION ORION
Similar to Starfleet Orion except single player only, different scenarios.
16K and 32K Pet versions on one tape $19.95

GAME PLAYING WITH BASIC TAPE 1, TAPE 2, AND TAPE 3
by Hayden .... $9.95 each

SoftSide Magazine the best software magazine for the Apple, TRS-80, and Atari computers. $18/year (12 issues)

SoftSide: Apple programs
on cassette $39.00 for 6 months
on diskette $69.00 for 6 months

Introductory Special!
Until Nov. 1. Magazine $15/yr. With cassette $69.95/yr. With disk $129.95/yr.

This is only a very small sample of our product line. For a complete selection, send $1 for our catalog of hardware, software and publications and receive a $2 credit toward your first order.

KEY	
I —	Integer BASIC
M —	Machine Language
A —	Applesoft
ROM -	Applesoft card or Apple II Plus only

The Software Exchange *6 SOUTH ST., MILFORD, NH 03055*

To order: Call Toll-Free 1-800-258-1790 *(in NH call (603) 673-5144)*

VISA

The Software Exchange & HardSide (Div. of Robitaille & Sons, Enterprises, Inc.), SoftSide Publications

Assembly Language Programming with UCSD PASCAL

J. M. Moshell

This article is primarily about using the APPLE II version of UCSD PASCAL to write 6502 assembly-language programs, and use them with PASCAL programs. Much of what is described can, however, be used with other UCSD PASCAL microcomputer systems. We include as an example a program to make low-resolution 16-color APPLE graphics available to PASCAL.

General Issues about assembler/PASCAL relations.

The UCSD system uses a very sophisticated " psuedo-machine" (P-machine) or interpreter program, which looks to the user like a sixteen-bit micro-computer with a "pure stack" architecture (no data registers); all operands are on the top of a stack, and the result of an operation is put back there. The UCSD system not only places data on the stack, it actually places executable code on the stack when it is brought in from disk. These chunks of code may either be P-machine code or 6502 code. Their interrelation is quite complex, and the APPLE PASCAL manual (essentially the UCSD PASCAL manual, customized for APPLE) does a reasonable job of explaining the operation of the system, assuming you have a graduate education in computer science. APPLE promises a new book in April, 1980. Here we're only interested in making you a fluent assembly-language user, so we'll skip most of the unnecessary details.

Basically, you may write any number of assembly-language subroutines, and link them together with a PASCAL "host program". Sometimes the host is trivial, doing nothing but calling your assembly-language program. This linkage is done by the LINKER program, and is not complicated to perform, if you have built the proper buttons and hooks into the host and the subroutines.

The examples in the APPLE book are simple and accurate. The .PROC pseudo-operator in your assembly program declares a name that can be used by the LINKER to make a connection with a procedure name that is declared EXTERNAL in the host program. Rather than repeating the APPLE manual we will concentrate on areas they don't stress.

Labels in an assembly program can receive their values in two ways: either by the EQUate pseudo-op, or by being attached to a line of code. In simple-minded absolute assemblers like that of CP/M, you can form complex expressions based on the assignment of value to a label; but the UCSD PASCAL assembler requires care in the construction of expressions. Any label, whose value is determined by being attached to a line of code, is a *relocatable* value. Its actual value will be assigned at run-time. Thus it cannot be used in forming any arithmetic expressions that involve more than adding a constant value to the relocatable value (e.g. LDA LABEL1 + 1 is o.k., and will load accumulator with the contents of the byte just after LABEL1). However, multiplying or dividing a relocatable value is illegal, since the run-time system hasn't got mechanisms for evaluating such expressions. For instance, in absolute assemblers, we would use something like:

```
    LDA #TABLE/256
    STA POINTER + 1
```

to pick up the high byte of TABLE's address, and

```
    LDA #TABLE MOD 256
    STA POINTER
```

to get its low byte.
With UCSD PASCAL, we have to use the following slightly more cumbersome technique:

```
    LDA TABADDR + 1; high-byte
    STA POINTER + 1
    LDA TABADDR ; low-byte
    STA POINTER
    . . . .
TABADDR: .WORD TABLE; store (relocatable)
address of table here.
TABLE: .BYTE 0, 2, 4, 15, 40
```

Now, the relocating loader will store the value of label TABLE at location TABADDR. You can then pick it to pieces by bytes, when your program runs.

The second way in which labels receive values is via the EQUate pseudo-op. The expression to the right of the EQU can contain absolute values freely intermixed with arithmetic and logical operators, but can only contain one relocatable expression to be added to the result of the absolute arithmetic. Thus, two legal EQUates are:

```
LABEL1 .EQU 80 ;reference to page zero,
location 80
LABEL 2 .EQU L5  + LABEL1/4
. . . . .
L5: .BYTE 44 ;L5 evaluated by its position in
the code file.
```

Another common source of confusion is that the UCSD Assembler strictly segregates code-generating operations (.PROC and .FUNC) from non-code-generating directives such as .EQU. Thus you cannot put an .EQU at any convenient place within a code-file; they must all occur before the *first* .PROC in your source file. The "high-level syntax" diagram in the assembler manual is worth some study.

"Local labels" such as $1 in this $1;

```
INC COUNTER
BNE $1
```

are useful but tricky. They are useful because they don't "last"; they are unknown except within a range of code delimited by two "real" labels; thus you can use $1 as a local label for short jumps anywhere within your program as long as another $1 isn't "within sight". The tricky part is that sometimes you will interpose, by accident, a real label during code modification, creating something like

```
$1 INC COUNTER
JUMPIN: DEC RECORD
        BNE $1
        . . .
NEXTLBL: etc
```

The assembler error message will occur at NEXTLBL, even though the error was the insertion of JUMPIN. The problem is that the one-pass assembler keeps on looking (within the region between JUMPIN and NEXTLBL) for a local $1 to jump to; it's been "cut off" from the one above JUMPIN. When NEXTLBL ends the region, an error results, far from its cause. Beware; always check that the jump and the local label can "see each other".

Communication with the PASCAL host program is well-described by the APPLE manual. Communication with other assembly routines is also easy, but there are a few twists. The pseudo-ops. .PROC and .DEF make any label to which they are attached, at the head of a routine, known to everyone else participating in linkage. Thus you can, in a separate source-file, use the .REF pseudo-op to make a label meaningful to *this* procedure, either for data access or for jumping to. For instance:

In one source-file:

```
    .DEF DATATABLE, FIXUP
    .PROC PART1
    . . .
DATATABLE: .BYTE 11, 40, 80, 0B0
    . . .
FIXUP: LDA ETC ;subroutine to be called from
               both within
               ;PART1 and from PART2.
```

In another source file:

```
    .REF DATATABLE, FIXUP
    .PROC PART 2
    . . . . .
    LDA DATATABLE + 1
    JSR FIXUP
    . . . .
```

You can put several .PROCs in one source file; this is convenient since that file becomes a kind of "library", and may be the only thing you have to link to your host. However, you *cannot* use REF and DEF between .PROCS in the same source file! In a strange way, a data-table in another source file is more accessible to some routine than one in its own source file.

Some Semi-Legal Communications Windows between PASCAL and Assembly Language

The "official" ways of communicating between assembly programs and PASCAL hosts are: procedure parameters and function values, and .PUBLIC and .PRIVATE entries in the global symbol table (that is, COMMON, to you FORTRAN types). These are well documented in the APPLE manual.

However, several "unofficial" ways exist for getting from one realm to the other. They rely on a sneaky trick which allows us to violate the type-rules of PASCAL; that is, the rules which separate integers from characters, etc.

We will use **pointer variables** in this discussion. Briefly, a pointer variable is a machine-memory address. In PASCAL they are widely used for linked-list management, and for many other things. If a pointer varible P is to be used as a "pointer to integers", then it is declared by

```
VAR P:^INTEGER;
```

and the expression

```
P^
```

refers to whatever location the pointer variable P is currently pointing at, viewed as a 16-bit integer. Thus, **Y: =^p** would pick up the integer value to which P points and put it into variable Y.

We could thus use pointers to make PEEKS and POKES if we just had a way of assigning an integer value to the pointer itself (the address we want to look at). However, PASCAL doesn't want to let us do that, because **pointer** is a different type than **integer.**

However, there exists a special declaration, which allows *two different* identifiers to refer to the *same* memory location. Thus, we will "trick" PASCAL into giving that location both a pointer-name and an integer name. We can then assign values to the integer name, and get at "what it points to". The code:

```
VAR X : RECORD
        CASE BOOLEAN OF
        TRUE: INT: INTEGER;
        FALSE: PTR:^INTEGER
        END;
```

I won't explain why this particular notation was used; but it works. Thus we can do a PEEK at location 1000, say, by making the assignments:

```
X.INT: = 1000;
Y: X.PTR^
```

Now Y contains the value that is in (decimal) locations 1000 and 1001 (low and high order bytes of a 16-bit word). We illustrate the trick with a PASCAL Function PEEK16:

```
FUNCTION PEEK16 (LOC:INTEGER):
INTEGER;
VAR X: RECORD
        CASE BOOLEAN OF
        TRUE: INT: INTEGER;
        FALSE: PTR: INTEGER
        END;
BEGIN
  X.INT: =LOC;
  PEEK: =X.PTR^
END;
```

A 16-bit POKE, and byte-sized PEEKS and POKES are supplied at the end of the article.

Now, for those of you who know something of pointer variables and dynamic storage allocation, you can use a POKE like the above, to store a pointer-value in some memory location (e.g. in page 0) then access it from assembly language to refer to the structure in dynamic storage to which the pointer points. (For those of you who don't use pointer variables, don't worry about it.)

Some specifically APPLE tricks for coding and debugging assembly language.

All the foregoing has been applicable to any UCSD PASCAL system. Let's look at some specific APPLE stuff.

First, the pseudo-machine interpreter in the APPLE is located at the high end of memory, in the RAM on the language-card. The pseudo-machine uses page zero memory locations 36 to A1 (hex). Locations 0-35 are temporaries; they are used by the P-machine and Turtlegraphics, but you may trash them. Locations A2-EF are "officially" reserved for the P-machine, but unused. I would recommend using them from the top down, since minor mods to the P-machine may come out later that use more space upwards from A2. Thus, data which you want to survive from one call of your assembly program to another can reside in the area A2-EF.

Second, PASCAL's use of memory (specifically, the dynamic heap) starts slightly above location 1000 (hex). To "free up" the memory region 2000-4000 for use as a high-resolution video buffer, one can use the following trick (with our "two-faced pointer", X, above).

```
    X.INT : =16384 ; (* =4000 hex *)
    RELEASE(X.PTR); (*Moves Heap to 4000*)
```

This should be done "first thing" in the main body of your PASCAL host program, if you're going to do high-resolution graphics. Of course, you can also do that by calling INITTURTLE; this is exactly what they do.

One of the most difficult things about programming in any computer's assembly language is debugging. With DDT in CP/M, or equivalent systems, you can "trace" a program, operation by operation, and see where it goes wrong (assuming you understand your own code.) In fact the APPLE ROM MONITOR used to contain trace and breakpoint facilities; but they were removed when the auto-start feature was added; so your APPLE with the Pascal Language card won't help much for debugging.

However, it *is* possible to "breakout" of an assembly program, use the Monitor to examine memory locations, and then resume operation of the program, even returning to the PASCAL host program. The main trick is to know how to turn off the Language Card's RAM memory, which is overlaying the ROM MONITOR. The following pieces of code are a small "macro library", which I include in the head of each of my assembly routines.

```
; DEBUGGING AIDS FOR APPLE ASSEMBLER
;
.MACRO MONWHERE ;store address of $1
JSR $1 ;at location (15, 06)
$1 PLA
STA 16 ;assembler assumes numbers are
hexidecimal
PLA
STA 15
.ENDM
.MACRO MONCALL ;store registers at
STA 12 ;locations (12, 13, 14);
STX 13
STY 14
STA 0C082 ;select the monitor ROM
JMP 0FF59 ;jump to the monitor
.ENDM
.MACRO RETPAS
STA 0C08B ;RESTORE RAM TO PRE-BREAK
;CONDITION.
RTS ;AND RETURN TO PASCAL.
.ENDM
```

MONWHERE is used to determine where in memory the code-file is loaded at present. Since JSR "pushes" the return address on the stack, we merely pull it off again and store it for examination by the monitor.

MONCALL stores the register contents, then enables the monitor-ROM and jumps to the monitor. Several monitor-calls can be put into a given assembly program and used almost like break-points. It is also possible, using the monitor, to put JMP instructions to the head of the MONCALL macro (JMP is easier to install than the whole thing, when you're working in hex-code).

RETPAS is used to return to a PASCAL host program. Since MONCALL turns off the Language Card RAM to get at the Monitor ROM, we must switch back to RAM to go back to PASCAL. To use RETPAS it is necessary to have saved the return address from the stack, before entering the ROM monitor. In the example program PUT40 (at the end of this article) we use an "interface sequence" to connect PASCAL to the assembly routine PLOT:

```
POP RETURN ; POP is a macro, stores return
addr. at RETURN.
POP YLOC ; Pull 16-bit param. arguments off
stack.
POP XLOC
POP COL
LDA COL ; Move low-order half to another place
STA COLOR ; in Page 0
JSR PLOT ; Go to actual routine
PUSH RETURN ; Put PASCAL return address
back on stack.
RTS ; and return to host program.
RETURN: .WORD 0
XLOC: .WORD 0
YLOC: .WORD 0
COL: .WORD 0
```

To modify and reassemble the routine for monitor breaks, one would remove RTS and insert the macro RETPAS. MONWHERE and MONCALL could be inserted at various points in PLOT for debugging purposes.

RETPAS will work whether or not a monitor call actually occurs; so it could be left in place if desired, for future debugging.

A useful plan for debugging an assembly/PASCAL program package is to put "debugging statements" in the PASCAL host, which print out all the variables. A READLN command is also inserted inside any loop which contains calls to the assembly routines. Thus, whenever you are in PASCAL carriage-return moves you through the loop, allowing time to note the values printed by the debugging statements. When you hit a MONCALL in the assembly routines, you can note the state of assembly-level variables by examining location (15, 16 -hex) to find the code's absolute address, then dumping memory (or deassembling it with the L command) to locate the memory variables. This combination of PASCAL-level and assembly-level debugging output is usually sufficient to debug anything you may write, if you are careful in the first place.

We will conclude with a "guided tour" of a simple program called QUILT.

QUILT: an example of PASCAL with assembler.
The QUILT program is a simple "pattern-repeating" program. A cursor color is selected with game-paddle 0. Typing keys around 'K' moves the cursor (e.g. typing 'I' moves it upward.) When you type 'R' the pattern is repeated, wrapping around the screen as it runs off an edge. This program is used as a first lesson in the concept of loops, in the high school computer science curriculum which we are developing.

Let's "walk through" QUILT; first, we'll examine the PASCAL code, then the assembly code.

PASCAL has a very rigid structure; you must declare constants, then variables, then procedures, then have the main program. Procedure EXPLAIN is used so as to place the explanation-text close to the beginning of the source program, to avoid having to repeat it in a block comment. Procedures C0L0R40 and PUT40 are declared as EXTERNAL, meaning that they will be attached later by the LINKER. They will be in assembly language.

Procedures PUTBUMP and FIXIT are convenient, since their task is needed several places in QUILT.

The main program consists of two parts: First, the input keystroke sequence is stored until 'R' is hit. Then, the sequence is replayed until any key is hit. We see that the main program begins with calls to procedures EXPLAIN and READLN. The call to READLN just "stalls" while you read the explanation, and hit a carriage return. COLOR40 turns on the APPLE in low-resolution mode. CLEAR40 clears the screen. We then go through a "case" statement to make entries in two arrays, XLIST and YLIST, which store changes in the X and Y position of the cursor.

When an 'R' is typed we fall out of the REPEAT.. UNTIL loop, and into another loop REPEAT... UNTIL KEYPRESS. This loop repeatedly goes through the XLIST and YLIST arrays, using them to move and display the colored cursor.

Finally, we have a call to TEXTMODE which is hit after the program detects 'Q' (quit). If this is not done you wind up back at the UCSD PASCAL monitor, but still in graphics mode; this is confusing.

Now let's look at the assembly routines:

Note first that we use .INCLUDE MACROS TEXT. We don't see the macros expanded here, but they are present when the file is assembled just as if typed in.

We see PROC COLOR40, which just hits the correct memory addresses to set the APPLE up in low-resolution mode with a 4-line text buffer still visible (for user instructions).

PROC PUT40 uses three parameters; these are "POP" ped off the stack and stored in memory words (since integers are pushed as 16-bit words by PASCAL). POP is another macro in MACROS. TEXT; see the end of this article for macros POP, PUSH and COPY, useful accessories.

The actual 40 x 40 plot routine is just copied from the ROM monitor; we could have actually jumped to it rather than including it here. It performs an address calculation to deal with the APPLE's strange way of mapping memory, puts the color in, and returns to the "interface" routine at the top. That routine pushes RETURN on the hardware stack and returns to PASCAL.

Good luck in writing assembly programs with PASCAL!

(Much thanks to Charlie Hughes and Al Hoffman for assistance and ideas.)

More commonly used macros:

```
.MACRO PDP
PLA
STA %1
PLA
STA %1+1
.ENDM
.MACRO PUSH
LDA %1+1
PHA
LDA %1
PHA
.ENDM
.MACRO COPY
LDY #%3 ; COPIES UP TO 256 BYTES
$1; LDA %1,Y ;USED AS FOLLOWS
STA %2,Y ;COPY SOURCE, DEST, LENGTH
DEY ;SOURCE, DEST ARE LABELS
BNE $1 ;LENGTH IS INTEGER,0-255
.ENDM ;0 MEANS COPY 256 BYTES.)
```

POKE and PEEK require the following declarations:

```
TYPE BYTE = 0..255; ARRAY8 = PACKED ARRAY [0..1]
OF BYTE;
VAR X: RECORD
        CASE BOOLEAN OF
        TRUE: (INT: INTEGER);
        FALSE: (PTR:^ INTEGER)
        END;
    Y: RECORD
        CASE BOOLEAN OF
        TRUE: (INT: INTEGER);
        FALSE: (PTR:^ARRAY8)
        END;
PROCEDURE PIKE16(WHAT,WHERE:INTEGER);
BEGIN
X.INT: = WHERE;
X.PTR^: = WHAT
END;
PROCEDURE POKE8(WHAT,WHERE:INTEGER();
BEGIN
Y.INT: = WHERE;
Y.PTR^[0]: = WHAT
END;
FUNCTION PEEK8(WHERE:INTEGER):INTEGER;
BEGIN
Y.INT: = WHERE;
PEEK8: = Y.PTR^[0]
END;
```

```
PROGRAM QUILT;
        USES APPLESTUFF,TURTLEGR;
        CONST SEEDSIZE=50;
        VAR  LISTPTR,X,Y,I,COL:INTEGER;
             XLIST.YLIST,CLIST:
             ARRAY[0..SEEDSIZE]OF INTEGER;
             INCHAR: CHAR;
PROCEDURE EXPLAIN;
  BEGIN
  WRITELN;
  WRITELN('THIS PROGRAM LETS YOU DRAW');
  WRITELN('A PICTURE AND THEN MAKE A');
  WRITELN(' ''QUILT'' OF IT BY REPEAT-');
  WRITELN('ING THE PATTERN. TO DRAW:');
  WRITELN;
  WRITELN('SET PADDLE 0 FOR A COLOR');
  WRITELN('TYPE KEYS AROUND K TO MOVE');
  WRITELN('THE DOT. FOR INSTANCE ''I'' ');
  WRITELN('MOVES THE DOT UPWARD.');
  WRITELN;
  WRITELN('TYPE ''R'' TO SEE THE PATTERN');
  WRITELN('REPEATED.');
  WRITELN;
  WRITELN('TYPE ''H'' TO HALT THE REPEATING.');
  WRITELN('TYPE ''C'' TO CLEAR SCREEN.');
  WRITELN('TYPE ''QUIT'' TO END PROGRAM.');
  WRITELN;
  WRITELN('NOW TYPE ''RETURN'' TO BEGIN.');
  WRITELN;
  WRITLEN('REMINDERS:');
  WRITELN('KEYS AROUND K:MOVE;I=UP,J=LEFT,ETC.');
  WRITELN('C)LEAR  R)EPEAT  H)ALT REPEATING  Q)UIT');
  END;
```

```
PROCEDURE COLOR40; EXTERNAL;
PROCEDURE PUT40(COLOR:INTEGER;
                XLOC:INTEGER;
                YLOC:INTEGER);EXTERNAL;
PROCEDURE CLEAR40;
      BEGIN
      FOR X:=0 to 39 DO
       FOR Y:=0 to 39 DO
        PUT40(0,X,Y);
      X:=20;Y:=20;
      END;
```

```
PROCEDURE PUTBUMP;
      BEGIN
      LISTPTR:=LISTPTR+1;
      IF LISTPTR=SEEDSIZE THEN LISTPTR:=0
      PUT40(COL,X,Y)
      END;
```

```
FUNCTION FIXIT(ARG:INTEGER):INTEGER;
      (*THIS IS NECESSARY BECAUSE
      UCSD'S 'MOD' FUNCTION DOESN'T
      WORK FOR NEGATIVE NUMBERS.DON'T
      REALLY KNOW WHY THAT IS.*)
      BEGIN
      FIXIT:=ARG;
      IF ARG>39 THEN FIXIT:=ARG-40;
      IF ARG<0 THEN FIXIT:=ARG+40;
      END;
```

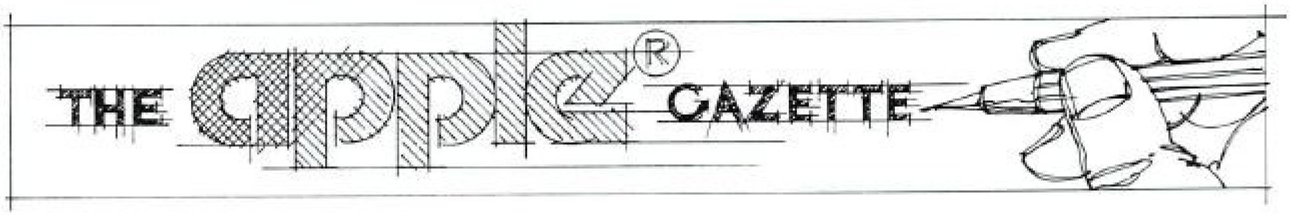
```
(* MAIN PROGRAM ----------------------
   ----------------------------------
   THIS CONSISTS OF TWO PARTS, THAT
   ARE REPEATED UNTIL SOMEONE TYPES 'Q':
   THE FIRST PART BUILDS UP AN IMAGE
   IN THE ARRAYS XLIST.YLIST.CLIST,
   WITH XLIST AND YLIST STORING THE
   'MOVES' -THAT IS. 0,+1 OR -1,THAT
   RESULT FROM EACH KEYSTROKE. CLIST
   IS A LIST OF COLORS THAT WERE IN
   EFFECT AT THE TIME OF EACH KEYSTROKE.
   THE SECOND PART RUNS THROUGH THE
   THREE LISTS AGAIN AND AGAIN, MOVING
   THE 'CURSOR' by ADDING XLIST[I]
   TO X, AND YLIST[I] TO Y,
   AND CHANGING THE COLOR 'COL' to
   CLIST[I].
   WE LET I RUN FROM 0 TO 'LISTPTR-1'
   BECAUSE LISTPTR WAS 'BUMPED' EVERY
   TIME A PICTURE ELEMENT (PIXEL) WAS
   ADDED; SO IT IS ONE TOO BIG.
   *)

     BEGIN
      (*SETUP STUFF*)
      EXPLAIN;
      READLN;
      COLOR40;
      CLEAR40;
      INCHAR:=' ';
      WHILE INCHAR<>'Q' DO
       BEGIN
      (* PART 1 : GET A PATTERN *)
      X:=20; Y:=20;
      LISTPTR:=0;
      REPEAT (* UNTIL 'R' *)
           REPEAT
           COL:=PADDLE(0) DIV 16;
           PUT40(COL,X,Y)
          UNTIL KEYPRESS;
          CLIST[LISTPTR]:=COL;
          READ(KEYBOARD,INCHAR);
          CASE INCHAR OF
          'U': BEGIN
                    XLIST[LISTPTR]:=-1;
                    X:=FIXIT(X-1);
```

```
        YLIST[LISTPTR]:=-1;
        Y:=FIXIT(Y-1);
        PUTBUMP
     END;
 'I': BEGIN
        XLIST[LISTPTR]:=0;
        YLIST[LISTPTR]:=-1;
        Y:=FIXIT(Y-1);
        PUTBUMP
     END;

 'O': BEGIN
        XLIST[LISTPTR]:=1;
        X:=FIXIT(X+1);
        YLIST[LISTPTR]:=-1;
        Y:=FIXIT(Y-1);
        PUTBUMP
     END;
 'J': BEGIN
        XLIST[LISTPTR]:=-1;
        X:=FIXIT(X-1);
        YLIST[LISTPTR]:=0;
        PUTBUMP
     END;
 'L': BEGIN
        XLIST[LISTPTR]:=1;
        X:=FIXIT(X+1);
        YLIST[LISTPTR]:=0;
        PUTBUMP
     END;

 'M': BEGIN
        XLIST[LISTPTR]:=-1;
        X:=FIXIT(X-1);
        YLIST[LISTPTR]:=1;
        Y:=FIXIT(Y+1);
        PUTBUMP
     END;

 ',': BEGIN
        YLIST[LISTPTR]:=1;
        Y:=FIXIT(Y+1);
        XLIST[LISTPTR]:=0;
        PUTBUMP
     END;

 '.': BEGIN
        XLIST[LISTPTR]:=1;
        X:=FIXIT(X+1);
        YLIST[LISTPTR]:=1;
        Y:=FIXIT(Y+1);
        PUTBUMP
     END;

      'C': CLEAR40;

      'Q': BEGIN
             TEXTMODE;
             EXIT(QUILT)
           END;
     END;(* CASE STATEMENT *)

UNTIL INCHAR='R';

  (* PART 2:

  NOW WE FALL OUT  OF THAT 'DRAWING'
  CODE (REPEAT BLOCK) AND FILL
  THE SCREEN WITH COPIES OF THE
  DRAWN IMAGE.*)
   REPEAT
     FOR I:=0 TO LISTPTR-1 DO
     BEGIN

       X:=FIXIT(X+XLIST[I]);
       Y:=FIXIT(Y+YLIST[I]);
       COL:=CLIST[I];
       PUT40(COL,X,Y)

     END
   UNTIL KEYPRESS;

   READ(KEYBOARD,INCHAR);

 END;(* BIG 'DO WHILE INCHAR<>'Q''
        BLOCK. WHEN Q IS HIT,
        PROGRAM ENDS. *)

 TEXTMODE;

END.
```

```
        ; MOSHELL'S IMPROVED 40X40
        ; PIXEL-PUTTER... HAS THREE
        ; ARGUMENTS (IN PASCAL ORDER:
        ;   COLOR, X-COORD,Y-COORD).
        ;   ALL 3 ARE OF TYPE INTEGER.
        ;
        ; THE ORIGIN IS IN UPPER LEFT
        ; CORNER; X IS HORIZONTAL, Y IS
        ; VERTICAL.
        ;
        ; THESE ROUTINES COME FROM THE
        ; APPLE ROM MONITOR. (IF I'D HAVE
        ; DONE IT I WOULD HAVE PUT THE
        ; ORIGIN IN LOWER LEFT...)
        ;
        ; THE FOLLOWING PAGE 0 LOCATIONS
        ; ARE JUST LIKE THOSE USED BY THE
        ; MONITOR. THEY ARE TEMPS, ONLY.
GBASL   .EQU     26
GBASH   .EQU     27
MASK    .EQU     2E
COLOR   .EQU     30
        ;
        ; THESE MACROS PROVIDE THE PUSH
        ; AND POP WE USE BELOW.
        .INCLUDE MACROS.TEXT
        ;
        ; THIS FIRST ROUTINE TURNS ON THE
        ; DISPLAY IN 40 X 40 COLOR MODE,
        ; WITH 4 LINE TEXT AT BOTTOM. TO
        ; GET 48 (VERT) X 40 (HORIZ), OMIT
        ; LAST LINE.
        .PROC    COLOR40
        STA      0C050    ;SELECT COLOR
        STA      0C054    ;PAGE 1
        STA      0C056    ;LO-RESOLUTION
        STA      0C053    ;TEXT @BOTTOM.
        RTS
        ;
        ;PUT40 TAKES 3 ARGUMENTS (SEE MAIN
        ;HEADER. THIS PRELIMINARY ROUTINE
        ;IS THE 'INTERFACE' TO PASCAL.
        ;
        .PROC    PUT40,3
        POP      RETURN
        POP      YLOC     ;WE POP STUFF OFF THE
        POP      XLOC     ;STACK in 16-BIT
        POP      COL      ;WORDS, THEN DISSECT
        LDA      COL      ;BY BYTES.
        STA      COLOR
        JSR      PLOT
        PUSH     RETURN
        RTS
RETURN  .WORD    0
XLOC    .WORD    0
YLOC    .WORD    0
COL     .WORD    0
        ;
        ;HERE IS THE PLOTTER.
        ;
PLOT:   LDA      YLOC
        LSR      A
        PHP
        JSR      GBASCALC; DO THE BIT PERMUTATIONS ...
        LDY      XLOC
        LDA      COLOR
        AND      #0F
        PLP               ;LO-BIT OF Y-ADDR
        BCC      LONYB    ;INTO CARRY.
      ;
      ; HIGH-NYB CASE (TOP HALF OF PIXEL)
      ;
        CLC
        ROL      A
        ROL      A
        ROL      A
        ROL      A        ;COLOR TO HI-NYB
        STA      COLOR
        LDA      #0F
WINDUP: AND      @GBASL,Y
        ORA      COLOR
        STA      @GBASL,Y
        RTS
LONYB:  STA      COLOR
        LDA      #0F0
        BCC      WINDUP   ;ALWAYS TAKEN.
        ;
        ;THE BASE-CALC:ROUTINE
        ;THIS DOES SOME BIT-PERMUTING...
        ;SEE THE NEW APPLE MANUAL FOR EXPLANATIONS.
        ;
        ;
```

```
GBASCALC PHA
         LSR      A
         AND      #03
         ORA      #04
         STA      GBASH
         PLA
         AND      #18
         BCC      GBCALC
         ADC      #7F
GBCALC   STA      GBASL
         ASL      A
         ASL      A
         ORA      GBASL
         STA      GBASL
         RTS
         .END
```

©



Randomize For The Apple II

Sherm Ostrowsky
291 Salisbury Avenue
Goleta, CA 93017

When you play games on your Apple II, do things start looking familiar after a while? Does the game get boring because you know just what is going to happen next, even though the random number generator in the program is supposed to make each event unexpected? There is a simple, one-line statement in BASIC which can remedy this problem; you can use it in your own programs, or even insert it into commercial programs after loading them.

The problem arises because the random numbers generated by the RND(1) function are part of a pseudo-random sequence which is always the same whenever you turn on the computer. You can select a different pseudo-random sequence by first entering a seed, S, and using a statement like:

```
X = RND(-S)
```

before any of the calls to RND(1). But this sequence, too, will always be the same very time you run your program with the same seed. What is really needed is a way to generate a starting seed which is different every time you run the program, and which is unknown to you.

Some versions of BASIC have a command ("RANDOMIZE") which does just this. Apple BASIC and APPLESOFT, unfortunately, do not have this command. A method which has been used by many Apple programmers to get around this difficulty is to ask for a starting seed from the user at the beginning of the program run, e.g.,

```
10 INPUT "SEED: ";S : X = RND(-S)
```

This will indeed start a different sequence of random numbers for that run, but it has some undesirable features. It may not be compatible with the ambience of the game ("Welcome to the space world of the twenty-third century' please enter a seed."). The user may not know enough about computers to understand what is wanted. And in any case, it is best if the seed is not known to the user, so each game can come as a complete surprise.

A somewhat more sophisticated approach, which I have seen used in at least one elegant program, uses a sequence such as

```
20 PRINT"HIT ANY KEY WHEN READY
 TO PROCEED";
30 X = RND(1): X = PEEK(-16384):
IF X < 128 GOTO 30
```

This does the job: while waiting for the key to be pressed, the repeated calls to RND place the system at an unknown and unpredictable location in the random number sequence. It is, however, not necessary to program this so directly, because the Apple monitor has a built-in routine which does the same thing.

The Apple's pseudo-RANDOMIZE function works as follows: whenever the cursor is blinking at you while awaiting some kind of input, a little machine-language loop is rapidly and repeatedly incrementing a two-byte integer stored at decimal locations 78 (low byte) and 79 (high byte). No matter how fast you respond by typing some input, this loop will have gone around so many times that the number stored in those locations will be quite random and unknown. Now, in order to get your program into the computer, it must obviously have been necessary either to type it in or read it in from tape or disk; either way, the blinking cursor must have appeared for you, even if only to await the LOAD command. Therefore, there will *always* be a random number waiting there to be your seed.

A simple way to use this pseudo-RANDOMIZE function is just to put the following statement near the beginning of your program:

```
10 X = RND(-PEEK(78) - 256 * PEEK(79))
```

Thereafter, any uses of RND(1) will get numbers out of a completely unpredictable sequence of random numbers. ©

SCREENDUMP

Jeff Schmoyer

Screendump is a machine language utility program which prints the contents of an Apple II text screen to any printer. It is executed by pressing a control-Z on the keyboard in response to any input. Its uses include printing the catalog without having to specifically start the printer, getting a hardcopy printout of instructions from programs, and selectively preserving information on the screen. Screendump will run on any size Apple II computer with or without a disk drive. It will work from Applesoft, Integer BASIC, or the Monitor.

Throughout this article control characters are printed in the format control-Z. This means press the Control key, and while holding it down, type Z or whatever other character is requested. Control characters are not displayed on the Apple's screen. All the addresses shown with dollar signs ($) in front are hexadecimal addresses.

When activated, Screendump replaces the system's standard character input hooks with its own. Normally when the computer wants a character, it goes to the Monitor keyboard input routine which waits for one to be typed and then passes it on. When Screendump is on, the computer goes to it for a character. It then checks to see if the character typed was a control-Z. If not, it passes it on just like the normal routine. If the character was a control-Z, it prints out the screen.

To accomodate different types of printers and interfaces, Screendump has its own output hooks at $2FE and $2FF. These should be set to the address of the printer driver routine which prints one character. On each of Apples' and most other manufacturers printer cards resides a ROM containing a printer driver routine to make the card work. After a PR#1 (if the card is in slot 1) is executed, the computer jumps to that driver whenever it wants to print a character. It does this by setting the systems output hooks to the appropriate driver address. For an Apple parallel card in slot 1 this address is $C102. If a different card is to be used, this address may be discovered through the following procedure.

Go to the Monitor by pressing Reset on a standard Apple II or by CALL -151 on an Apple II Plus or an Apple II containing a Language System.

Type control-P control-K and Return to disconnect the DOS.

Type the slot number that the printer card is in followed by a control-P and Return. For slot 1 this would be 1 control-P Return.

Type 36.37 Return. This will display the printer address in reverse order. For $C102 it would show 36:02 C1.

These need to be placed in Screendump in reverse order also. For $C102, $2FE should get $02 and $2FF should get $C1.

One other change is required for systems without disk drives. The value at locations $2B7 needs to be changed from $4C to $60.

To *use* Screendump from the Monitor type 2AFG and Return or from either BASIC type CALL 687 and Return. Now any time control-Z is pressed, the current screen will be printed. Screendump may be used anytime up to the next Reset or IN# command.

To *save* Screendump to tape, from the Monitor type 2AF,2FFW. To save it to disk type BSAVE SCREENDUMP,A$2AF,L$51. To reload it from tape, go to the Monitor and type 2AF.2FFR. To reload it from disk enter BLOAD SCREENDUMP or alternatively BRUN SCREENDUMP. The latter would both load Screendump and activate it.

Some printer cards, such as Apples' parallel card, need to be initialized before they can be used. This initialization must be done each time the computer is turned on, or with some cards, each time Reset is pressed. For a parallel card in slot 1 the sequence would be

PR#1
control-I 40N control-I K
PR#0.

After Screendump is activated, through the execution of a CALL 687 or one of the other previously described turn-on procedures, it may be utilized by typing a control-Z in response to any input. For example if the catalog is being displayed and the computer is waiting for any key to be pressed before showing more, control-Z can be entered and what is on the screen will be printed. The catalog will not advance until some other key is pressed. As another example, assume you are playing your favorite adventure game and it is waiting for you to enter a command. Typing control-Z will print the current screenful of information describing your whereabouts for future reference.

In some cases the control-Z character may need to be used by other programs or devices such as the Micromodem II, for their own purposes. If Screendump is active, it will never let a control-Z go through he system. To make a different use of control-Z, Screendump may be deactivated through an IN#0 or Reset, or the Screendump execution character can be changed to something other than a control-Z. This is accomplished from Applesoft by typing POKE 702,CHR$(''newchar''). The character in quotes, newchar, may be any character the system is not using for something else. For example, an A would not be a good character to use since everytime an A was typed the printer would start.

The operation of Screendump is as follows. SDINIT is the startup routine. It takes SCREENDUMPs address and puts it in the input hooks so that Screendump is called to get each character. If a disk is being used, the DOS is jumped to, passing the input address information along to it.

As previously outlined, when the system wants a

16K Ram Expansion Board for the Apple II*

$195.00

- expands your 48K Apple to 64K of programmable memory
- plugs in slot 0
- allows you to run Apple's new Fortran package and other languages
- eliminates the need for an Applesoft* or Integer Basic ROM Card
- switch selection of RAM or mother board ROM language
- includes installation and use instructions

Visa & Master Charge accepted
Shipping & handling will be added unless the order is accompanied by a check or money order
NC residents add 4% sales tax
*Apple II and Applesoft are trademarks of Apple Computer, Inc.
Allow 4-6 weeks for delivery

ANDROMEDA
COMPUTER SYSTEMS
P.O. Box 19144
Greensboro, NC 27410
(919) 852-1482

character it goes to SCREENDUMP which in turn looks to the Monitor routine KEYIN for a character. It then checks to see if the character entered was a control-Z. If not, it returns the character to the caller, your adventure program or whatever.

DUMPIT is the routine that actually prints the screen. First it saves the CPU registers and the current screen pointers. This is so that when it is finished, the cursor position and other information will still be intact.

Next it zeros the X register which will serve as the screen line counter. The Y register will contain the character position on that line. The X register (the line) is then transferred to the accumulator (the A register) and the Monitor routine GBASCALC is called. This routine translates the line number in the accumulator into the actual location of where that line starts in computer memory.

The forty characters for that line are now printed followed by a carriage return. The routine then goes to the next line and so on until it has done 24 lines.

After it finishes the screen, it restores the saved screen pointers and registers. Finally it goes to get a new character. It does this instead of passing the control-Z on to the system and causing a probable SYNTAX ERROR.

Screendump resides in memory at the end of the keyboard buffer area. These locations are generally not used by other programs but if a very long line is typed in, over 170 characters, Screendump will be destroyed. If it was active at the time of destruction, the computer will stop or do strange things. Hit Reset to recover.

```
:ASM
0000:             ; SCREENDUMP
0000:             ;
0000:             ;  DUMP SCREEN TO PRINTER WHENEVER
0000:             ;  CONTROL-Z IS PRESSED.
0000:             ;
0000:             ;     BY JEFF SCHMOYER  5/80
0000:             ;
0000:             ;
0000:             KSWL        EQU $38          CHAR IN HOOKS
0000:             KSWH        EQU KSWL+1
0000:             DOSSET      EQU $3EA         DOS SET HOOK ROUTINE
0000:             CH          EQU $24          CURSOR HORIZONTAL
0000:             GBASL       EQU $26          BASE LINE ADDRESS
0000:             GBASH       EQU GBASL+1
0000:             GBASCALC    EQU $F847        CALCULATE BASE ADDRESS ROUTINE
0000:             PRINT       EQU $C102        PRINTER CARD CHAR OUT ADDRESS
0000:             RDKEY       EQU $FD0C        MONITOR IN
0000:             KEYIN       EQU $FD1B        GET ONE PRESS
0000:             CR          EQU $8D          CARRIAGE RETURN
0000:             ;
02AF:                         ORG $2AF
02AF:                         OBJ $2AF
02AF:             ;
02AF:             ;  INITIALIZE THE INPUT HOOKS
02AF:             ;  TO POINT TO OUR ROUTINE.
02AF:             ;
02AF: A9 BA       SDINIT      LDA #SCREENDUMP
02B1: 85 38                   STA KSWL
02B3: A9 02                   LDA #SCREENDUMP/256
02B5: 85 39                   STA KSWH
02B7: 4C EA 03                JMP DOSSET       MOVE SCREENDUMP INPUT HOOKS TO DOS
02BA:             ;
02BA:             ;  GET A CHAR FROM THE KEYBOARD
02BA:             ;  AND CHECK FOR CONTROL-Z.
02BA:             ;  IF NOT THEN RETURN CHAR TO CALLER.
02BA:             ;
02BA: 20 1B FD    SCREENDUMP  JSR KEYIN        GET A CHAR
02BD: C9 9A                   CMP #$9A         IS IT A CTRL-Z?
02BF: F0 01                   BEQ DUMPIT       YES DUMP SCREEN
02C1: 60                      RTS              NO, SEND BACK CHARACTER
```

```
02C2:              ;
02C2:              ;   SAVE CURRENT POINTERS AND
02C2:              ;   PRINT THE SCREEN.
02C2:              ;
02C2: 8A           DUMPIT      TXA                 SAVE REGS
02C3: 48                       PHA
02C4: 98                       TYA
02C5: 48                       PHA
02C6: A5 26                    LDA GBASL           SAVE CURRENT SCREEN POINTERS
02C8: 48                       PHA
02C9: A5 27                    LDA GBASH
02CB: 48                       PHA
02CC: A5 24                    LDA CH
02CE: 48                       PHA
02CF: A2 00                    LDX #0              LINE COUNTER
02D1: 86 24                    STX CH              ZERO CURSOR HORIZONTAL
02D3: A0 00        NEXTLINE    LDY #0              COLUMN COUNTER
02D5: 8A                       TXA                 A GETS LINE
02D6: 20 47 F8                 JSR GBASCALC        TRANSLATE IT
02D9: B1 26        NEXTCHAR    LDA (GBASL),Y       GET A CHAR
02DB: 20 FD 02                 JSR PRINTONE        OUT WITH IT
02DE: C8                       INY                 MOVE TO NEXT CHAR
02DF: C0 28                    CPY #40             LINE DONE?
02E1: D0 F6                    BNE NEXTCHAR        NO
02E3: A9 8D                    LDA #CR
02E5: 20 FD 02                 JSR PRINTONE
02E8: E8                       INX                 NEXT LINE
02E9: E0 18                    CPX #24             ALL DONE?
02EB: D0 E6                    BNE NEXTLINE        NO
02ED: 68                       PLA                 PUT OLD LINE STUFF BACK
02EE: 85 24                    STA CH
02F0: 68                       PLA
02F1: 85 27                    STA GBASH
02F3: 68                       PLA
02F4: 85 26                    STA GBASL
02F6: 68                       PLA
02F7: A8                       TAY                 RESTORE REGS
02F8: 68                       PLA
02F9: AA                       TAX
02FA: 4C 0C FD                 JMP RDKEY           GET NEW KEYPRESS
02FD:              ;
02FD:              ;   JUMP TO ACTUAL PRINTER DRIVER
02FD:              ;   CHARACTER OUTPUT ROUTINE.
02FD:              ;
02FD: 4C 02 C1     PRINTONE    JMP PRINT

      0   ERRORS IN THIS ASSEMBLY
```

©

Some routines from Applesoft Basic

Jim Butterfield, Toronto

Routines were identified by examining specific memory dumps. There may well be other versions of Basic; the user is urged to exercise caution.

The addresses given identify the start of the area in which the described routine lies. This may not be the proper program entry point or calling address.

DISK	ROM	Description
0800	D000	Action addresses for primary keywords
0880	D080	Action addresses for functions
08B2	D0B2	Hierarchy and action addresses for operators
08D0	D0D0	Table of Basic keywords
0A60	D260	Basic messages, mostly error messages
0B65	D365	Search the stack for FOR or GOSUB activity
0B93	D393	Open up space in memory
0BD6	D3D6	Test: stack too deep?
0BE3	D3E3	Check available memory
0C10	D410	Send canned error message, then:
0C3C	D43C	Warm start; wait for Basic command
0C5C	D45C	Handle new Basic line input
0D0F	D50F	Rebuild chaining of Basic lines
0D2E	D52E	Receive line from keyboard
0D59	D559	Crunch keywords into Basic tokens
0E1A	D61A	Search Basic for given line number
0E49	D649	Perform NEW
0E6A	D66A	Perform CLEAR
0E99	D697	Reset Basic execution to start
0EA7	D6A5	Perform LIST
0F68	D766	Perform FOR
102A	D828	Execute Basic statement
104B	D849	Perform RESTORE
1070	D86E	Perform STOP or END
1098	D896	Perform CONT
10B2	D8B0	Perform SAVE
10CB	D8C9	Perform LOAD
1114	D912	Perform RUN
1123	D921	Perform GOSUB
1140	D93E	Perform GOTO
116D	D96B	Perform RETURN/POP, then:
1197	D995	Perform DATA: skip statement
11A5	D9A3	Scan for next Basic statement
11A8	D9A6	Scan for next Basic line
11CB	D9C9	Perform IF, and perhaps:
11DE	D9DC	Perform REM: skip line
11EE	D9ED	Perform ON
120E	DA0C	Input fixed-point number
1248	DA46	Perform LET
12D1	DACF	Perform PRINT
133D	DB3A	Print string from memory
135A	DB57	Print single format character
1374	DB71	Handle bad input data
13A3	DBA0	Perform GET
13B5	DBB2	Perform INPUT
13E5	DBE2	Perform READ
14E2	DCDF	Canned Input error messages
14FC	DCF9	Perform NEXT
1558	DD55	Check type mismatch
157E	DD7B	Evaluate expression
16B5	DEB2	Evaluate expression within parentheses
16BB	DEB8	Check parenthesis, comma
16CC	DEC9	Syntax error exit
16D8	DED5	Setup for variables
1713	DF10	Set up function references
1752	DF4F	Perform OR, AND
1768	DF65	Perform comparisons
17D0	DFCD	Perform PDL
17DC	DFD6	Perform DIM
17E6	DFE3	Get variable name, location
18E6	E0ED	Setup array pointer
18FB	E102	Evaluate integer expression
1917	E11E	Find or make array
1AD7	E2DE	Perform FRE, and:
1AEB	E2F2	Convert fixed-to-floating
1AF8	E2FF	Perform POS
1AFF	E306	Check not Direct
1B0C	E313	Perform DEF
1B3A	E341	Check FNx syntax
1B4D	E354	Evaluate FNx
1BBE	E3C5	Perform STR$
1BCC	E3D5	Do string vector
1BDE	E3E7	Scan, set up string
1C49	E452	Build descriptor
1C7B	E484	Garbage collection
1D8E	E597	Concatenate
1DCB	E5D4	Store string
1DF4	E5FD	Discard unwanted string
1E2C	E635	Clean descriptor stack
1E3D	E646	Perform CHR$
1E51	E65A	Perform LEFT$
1E7D	E686	Perform RIGHT$
1E88	E691	Perform MID$
1EB0	E6B9	Pull string data
1ECD	E6D6	Perform LEN
1ED3	E6DC	Switch string to numeric
1EDC	E6E5	Perform ASC
1EEC	E6F5	Get byte parameter
1EFE	E707	Perform VAL
1F3D	E746	Get two parameters for POKE or WAIT
1F49	E752	Convert floating-to-fixed
1F5B	E764	Perform PEEK
1F72	E77B	Perform POKE
1F7B	E784	Perform WAIT
1F97	E7A0	Add 0.5
1F9E	E7A7	Perform subtraction
1FB0	E7B9	Perform addition
2095	E89E	Complement accum#1
20CC	E8D5	Overflow exit
20D1	E8DA	Multiply-a-byte
210A	E913	Constants
2138	E941	Perform LOG
2179	E982	Perform multiplication
21DA	E9E3	Unpack memory into accum#2
2205	EA0E	Test & adjust accumulators
2222	EA2B	Handle overflow and underflow
2230	EA39	Multiply by 10
2247	EA50	10 in floating binary
224C	EA55	Divide by 10
2257	EA60	Perform divide-by
225D	EA66	Perform divide-into
22F0	EAF9	Unpack memory into accum#1
2315	EB1E	Pack accum#1 into memory
234A	EB53	Move accum#2 to #1
235A	EB63	Move accum#1 to #2
2369	EB72	Round accum#1
2379	EB82	Get accum#1 sign
2387	EB90	Perform SGN
23A6	EBAF	Perform ABS
23A9	EBB2	Compare accum#1 to memory
23E9	EBF2	Floating-to-fixed
241A	EC23	Perform INT
2441	EC4A	Convert string to floating-point
24CC	ECD5	Get new ASCII digit
2501	ED0A	Constants
2510	ED19	Print IN, then:
2517	ED20	Print Basic line #
252B	ED34	Convert floating-point to ASCII
265B	EE64	Constants

2684	EE8D	Perform SQR
268E	EE97	Perform power function
26C7	EED0	Perform negation
26D2	EEDB	Constants
2700	EF09	Perform EXP
2753	EF5C	Series evaluation
279D	EFA6	RND constants
27A5	EFAE	Perform RND
27E1	EFEA	Perform COS
27E8	EFF1	Perform SIN
2831	F03A	Perform TAN
285D	F066	Constants
2895	F09E	Perform ATN
28C5	F0CE	Constants
2902	F10B	CHRGET sub for zero page
291F	F128	Basic cold start
29DC	F1D5	Perform CALL
29E5	F1DE	Perform IN#
29EC	F1E5	Perform PR#
2A2C	F225	Perform PLOT
2A39	F232	Perform HLIN
2A48	F241	Perform VLIN
2A56	F24F	Perform COLOR=
2A5D	F256	Perform VTAB
2A69	F262	Perform SPEED=
2A74	F26D	Perform TRACE, NOTRACE
2A7A	F273	Perform NORMAL, INVERSE
2A87	F280	Perform FLASH
2A8D	F286	Perform HIMEM:
2AAD	F2A6	Perform LOMEM:
2AD2	F2CB	Perform ONERR:
2B1F	F318	Perform RESUME
2B38	F331	Perform DEL
2B8C	F390	Perform GR
2B95	F399	Perform TEXT
2B9B	F39F	Perform STORE
2BB8	F3BC	Perform RECALL
2BD4	F3D8	Perform HGR2, HGR
2C0D	F411	Varous graphics subroutines
2EE5	F6E9	Perform HCOLOR=
2EFA	F6FE	Perform HPLOT
2F1A	F721	Perform ROT=
2F20	F727	Perform SCALE=
2F62	F769	Perform DRAW
2F68	F76F	Perform XDRAW
2F6E	F775	Perform SHLOAD
2FE0	F7E7	Perform HTAB

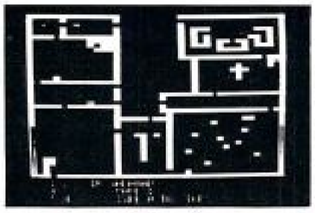
Applesoft memory map (Page O)

Hex	Decimal	Description
000D	13	Search character
000E	14	Scan-between-quotes flag
000F	15	Input buffer pointer; # of subscripts
0010	16	Defa ult DIM flag
0011	17	Type: FF=string, 00=numeric
0012	18	Type: 80=integer, 00=floating point
0013	19	Flag: DATA scan; LIST quote; memory
0014	20	Subscript flag; FNX flag
0015	21	0=INPUT; $40=GET; $98=READ
0016	22	Comparison Evaluation flag
0024	36	Position on print line
0050-0051	80-81	Integer value (for GOTO etc)
0052-0054	82-84	Pointers for descriptor stack
0055-005D	85-93	Descriptor stack(temp strings)
005E-0061	94-97	Utility pointer area
0062-0066	98-102	Product area for multiplication
0067-0068	103-104	Pointer: Start-of-Basic
0069-006A	105-103	Pointer: Start-of-Variables
006B-006C	107-108	Pointer: Start-of-Arrays
006D-006E	109-110	Pointer: End-of-Arrays
006F-0070	111-112	Pointer: String-storage(moving down)
0071-0072	113-114	Utility string pointer
0073-0074	115-116	Pointer: Limit-of-memory
0075-0076	117-118	Current Basic line number
0077-0078	119-120	Previous Basic line number
0079-007A	121-122	Pointer: Basic statement for CONT
007B-007C	123-124	Current DATA line number
007D-007E	125-126	Current DATA address
007F-0080	127-128	Input vector
0081-0082	129-130	Current variable name
0083-0084	131-132	Current variable address
0085-0086	133-134	Variable pointer for FOR/NEXT
0087-008F	135-143	Work area, pointers, etc
0090-0092	144-146	Jump vector for functions
0093-009C	147-156	Misc numeric work area
009D	157	Accum#1: Exponent
009E-00A1	158-161	Accum#1: Mantissa
00A2	162	Accum#1: Sign
00A3	163	Series evaluation constant pointer
00A4	164	Accum#1 hi-ordeer (overflow)
00A5-00AA	165-170	Accum#2: Exponent, etc.
00AB	171	Sign comparison, Acc#1 vs #2
00AC	172	Accum#1 lo-order (rounding)
00AD-00AE	173-174	Series pointer
00B1-00C8	177-200	CHRGET subroutine; get Basic char
00B7	183	Sub entry: get prev character
00B8-00B9	184-185	Basic pointer (within subrtn)
00C9-00CD	201-205	Random number seed.
0200-02FF	512-767	Input buffer

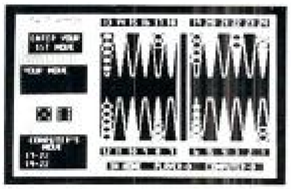
©

Exciting, entertaining software for the Apple II and Apple II Plus*

If you liked "Invaders", you'll love ASTEROIDS IN SPACE by Bruce Wallace. Your space ship is traveling in the middle of a shower of asteroids. Blast the asteroids with lasers, but beware — big asteroids fragment into small asteroids! The Apple game paddles allow you to rotate your space ship, fire its laser gun, and give it thrust to propel it through endless space. From time to time you will encounter an alien space ship whose mission is to destroy you, so you'd better destroy it first! High resolution graphics and sound effects add to the arcade-like excitement that this program generates. Runs on any Apple II with at least 32K and one disk drive.
On diskette — $19.95

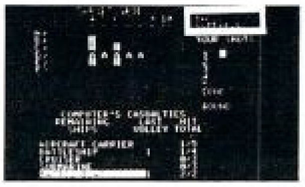


FRACAS™ by Stuart Smith. A fantastic adventure game like no other — up to eight players can participate in FRACAS at the same time. Journey in the land of FAROPH, searching for hidden treasure while warding off all sorts of unfriendly and dangerous creatures like the Ten Foot Spider and the Headless Horseman. You and your friends can compete with each other or you can join forces and gang up on the monsters. Your location is presented graphically and sound effects enliven the battles. Save your adventure on diskette or cassette and continue it at some other time. Requires at least 32K of RAM. Cassette: $19.95 Diskette: $24.95



FASTGAMMON™ by Bob Christiansen. Sound, hi res color, and cartoons have helped maked this the most popular backgammon-playing game for the Apple II. But don't let these entertaining features fool you — FASTGAMMON plays serious backgammon. Requires at least 24K of RAM.
Cassette: $19.95 Diskette: $24.95

BATTLESHIP COMMANDER™ by Erik Kilk and Matthew Jew. A game of strategy. You and the computer each start out by positioning five ships of different sizes on a ten by ten grid. Then the shooting starts. Place your volleys skillfully — a combination of logic and luck are required to beat the computer. Cartoons show the ships sinking and announce the winner. Sound effects and flashing lights also add to the enjoyment of the game. Requires at least 32K of RAM.
Cassette: $14.95 Diskette: $19.95



QS QUALITY SOFTWARE

6660 Reseda Blvd., Suite 105, Reseda, CA 91335

WHERE TO GET IT: Call us at (213) 344-6599 for the name of the Quality Software dealer nearest you. If necessary, you may order directly from us. MasterCharge and Visa cardholders may place orders by telephone and we will deduct $1 from orders over $19 to compensate for phone charges. Or mail your order to the address above. California residents add 6% sales tax. SHIPPING CHARGES: Within North America orders must include $1.50 for first class shipping and handling. Outside North America the charge for airmail shipping and handling is $5.00 — payable in U.S. currency.

*"Apple II" and "Apple II Plus" are trademarks of Apple Computer, Inc.

Al Baker's Programming Hints: Apple

Different computers confront the software designer with different problems. The most difficult task in the design process is making efficient use of a computer's assets while avoiding its liabilities. Often, a good design will convert a potential liability into an asset. Far more often, poor design will accentuate a liability. In this case, the software user is left with the results and must live with the problem or go elsewhere.

Five potential design problems with the Apple II immediately come to mind. These are:

No lower case.

Only 40 characters per line on the screen.

No mixed graphics and text.

Missing up-arrow and down-arrow keys.

No Joysticks.

If you own another computer, don't feel superior. I'm sure you have your own list of "Why did they do it that way?"

Look at the list. If you've bought much Apple software, you can probably think of products which successfully bypass, or even capitalize, on these "liabilities". The exception is the lack of Joysticks.

Why Joysticks?

Some software designs demand the use of joysticks. A joystick gives the user instantaneous control over direction of motion. Move the joystick left and the object on the screen instantly starts moving left. Let go of the joystick and it stops. Push the joystick to the right and the object immediately begins moving right.

A joystick has a natural center. Motion away from center is obvious, easy to program, and natural for the user. This control doesn't come naturally to a paddle. The paddle has no natural center. Everything is relative. One approach often used is to divide the paddle's turning radius into thirds: left, center, and right. This normally doesn't work because the user gets lost in the action of the game and loses track of where the center region is.

Most software designers give up and convert the values returned by the paddle into absolute positions on the screen. This is the most unsatisfactory solution of all. The user has lost instantaneous control of screen motion and is often left with a frustrating playing experience. If he has played the same game elsewhere, he must learn a new set of reflexes -- or give up.

The best example of this is the game SPACE INVADERS. The official MIDWAY arcade and Atari home versions of this game use joysticks, or the equivalent, to move the gun. The player has instantaneous control over the gun motion and can spend his psychic energies trying to shoot while avoiding enemy fire. The Apple versions I have played do not provide this level of user control. The programmers chose to use paddle values as absolute gun locations and the user is forced to deal with a gun that seems to have a life of its own. It is always moving in an attempt to reach the screen location that matches the paddle.

Paddle as Joystick

In the listing is an Applesoft example of a joystick simulation routine using the Apple II paddle. Type in the program and run it. You will see a snake made of X's running down the center of the screen. To control the snake, grab the paddle and yank it to the left -- either a lot or a little, and then yank it to the right. The snake started moving left and then stopped. Now yank the paddle right and then left. The snake started moving right and then stopped. instantaneous control that feels right!

The joystick simulator had several major design constraints. First, the center of the paddle must be umimportant. Second, how much the user turns the paddle has to be ignored, as long as it is enough to register. It has to be OK for the paddle to jitter with no effect. Third, the speed the user turns the paddle has to be ignored, as long as it is fast enough to register. If the snake is moving left and the player turns the joystick right slowly for most of its radius, the snake should stop, but it shouldn't stop and then move right. To do this, the user must turn the paddle to stop the snake and then turn the paddle again to start the snake moving right -- yank right, yank right.

Substituting this routine in SPACE INVADERS would recreate the proper "feel" of joystick control that is now missing.

The Program

Lines 1000 to 1080 form the body of the joystick simulator with JY as the value of the joystick. JY can have three values: -1 if the joystick is pushed left, 0 if it is centered, and +1 if it is pushed right. To properly simulate the joystick, the routine needs two facts: whether or not the paddle has been moving and which direction it is moving now. If the paddle was previously moving, then the user is still in the middle of yanking the paddle and the routine must ignore his input. If the paddle was still, then the routine should change the value of JY based on the current paddle motion.

Two variables are used to compute the joystick's current motion. PA is the value of the paddle now and PM is the value the paddle had last time. PA is obtained in line 1000. In line 1070, PM is assigned the value of PA prior to the RETURN.

PM is also one of the two variables used to compute the paddle's previous motion. PG is the other and always contains the value of PA from two times ago. Line 1060 sets up PG from the value of PM.

Let's follow the routine. Line 1000 obtains the current paddle value and line 1010 determines if the paddle was moving last time. This is determined by looking at the difference between PM and PG. If it was moving, the routine ignores the paddle but updates historic data beginning at line 1060.

```
10  REM        THE PADDLE AS JOYSTICK
20  REM
30  REM
40 PM =  PDL (0)
50 H  = 20
60  REM
70  REM          DEMO LOOP
80  REM
90  GOSUB 1000
100  PRINT  TAB( H)"X"
110 H  = H + JY
120  IF  H > 39 THEN H = 39
130  IF H < 1 THEN H = 1
140  GOTO 90
910  REM
920  REM
930  REM        JOYSTICK SIMULATION
940  REM
950  REM  JY=JOYSTICK READING
960  REM  PA=CURRENT PADDLE READING
970  REM  PM=MOTHER PADDLE READING
980  REM  PG=GRANDMOTHER PADDLE READING
990  REM
1000  PA =  PDL (0)
1010  IF  ABS (PM - PG) > 20 THEN 1060
1020  IF  ABS (PA - PM) < 20 THEN 1060
1030  IF PA < PM THEN JY = JY - 1
1040  IF PA > PM THEN JY = JY + 1
1050 JY =  SGN (JY)
1060 PG = PM
1070 PM = PA
1080  RETURN
```

Line 1020 ignores the paddle if it is only jittering or is being moved very slowly. If the paddle is moving quickly enough, lines 1030 and 1040 update the joystick value by subtracting 1 if it is turning counterclockwise and adding 1 if it is turning clockwise. Finally, line 1050 keeps JY within the range -2 to +1 and lines 1060 and 1070 update historic paddle data.

The program between lines 10 and 140 tests the joystick simulator with the moving snake discussed earlier. Lines 40 and 50 set up the historic joystick data and positions the head of the snake. After calling the joystick routine and moving the snake forward in lines 90 and 100, the position of the head is changed by the position of the joystick in line 110. Lines 120 and 130 keep the snake on the screen.

Conclusion

We've explored the Apple paddle and looked at various ways to use it. If you have other ways of using the paddles or improvements on what I have discussed, please send them to me. I will give full credit for anything I use. Also, I am interested in any ideas you have or specific problems you would like explained.

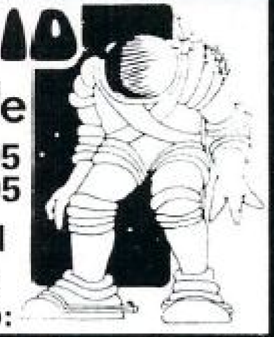
Al Baker Programming Director The Image Producers, Inc.
615 Academy Dr. Northbrook, IL 60062 ©

NEW FOR APPLE 2 & APPLE 2 Plus

ASTEROID

A REAL TIME HI-RESOLUTION GRAPHICS GAME WITH SUPER SOUND EFFECTS. THREE SKILL LEVELS- NOVICE, AVERAGE ("Arcade" difficulty") and EXPERT WHERE ASTEROIDS ARE ACTUALLY ATTRACTED TO YOUR SHIP!

ASTEROID



Now Available

32K TAPE -$14.95
32K DISK -$19.95

From Your Local Computer Store Or Send Order to:

©Copyright 1980
Adventure INTERNATIONAL
A DIVISION OF SCOTT ADAMS, INC

BOX 3435
LONGWOOD FLA 32750
(305) 862-6917

DEALER INQUIRIES INVITED

A Model for Structured Programming

The Anatomy Of A Word-Research Processing Program for the APPLE

Derek A. Kelly

Sophisticated and broadly-ranging readers will hopefully forgive the biological analogy in the title. While we most usually look upon programs and APPLEs as simply robot-like physical mechanisms, I've found that I can't regard programs simply in terms of the static analogies of machines, but that I must also regard them in biological terms, terms more suitable to living creatures and their structures which may evolve over time. Computer programs and programming follow a recursive pattern, generally, in that the structure of a program, its design, and its joints, is never finished, but is constantly changing in response to new situations and requirements.

One of the aims of this article is to present a sort of broadly ranging tutorial on programming and system design. Another aim is to discuss the structure--and some of the programming features--of a program that I use frequently, and which other research-oriented, word processing, scholars, undergraduate or graduate, may also find to be useful. Since my program is not "finished" in the sense that while every routine now in it works, I am not satisfied with the routines I selected for inclusion, and am constantly adding new ones. So I'll be talking about the "anatomy" of a growing and evolving program, not about a static and finished product. After having explained the problem and design of the program, I think a new programmer will be able to take up where I left off, and either code the entire program by himself, or he will be armed with the tools with which to design and program his own version of this program.

By word-research, I mean simply research carried out by reading & studying books containing mainly verbal information.

Problem Definition and Analysis

Anyone who wishes to prepare programs in a "structured" manner will need to follow the steps of this article, or ones quite similar. The first step in program construction is not--unless only a short, experimental program is desired--to turn on the APPLE and begin to code in lines, but to think out the program ahead of time, in the head, and on paper. This thinking out ahead of time is not just a short-lived matter. It may take up to 50 or 60% of your programming time, leaving the rest of the time for actual coding and debugging.

The first decision I must make when constructing a program is: What does this program do? What is the goal of the program? What problem does this program solve? What practical applications does it have? What results do I want to achieve from a use of such a program?

Let me explain the alternatives and the decisions on these questions that I made for the program of this article.

Millions of college students--and a proportionate number of professors--periodically face the prospect of writing a "paper" based on original research, for various courses in the Humanities and Social Sciences, and less frequently in the Physical Sciences.

The papers written to report on research projects all share a common set of characteristics on the formal level: all include an alphabetized bibliography, a series of footnotes, basic divisions, and a semi-standard format. A computer program could assist the student in conceiving a research project, organizing and designing a research report, assist him in gathering references and bibliography, and finally, can be of help to the writer in those final, frantic, hours when some order and organization must be imposed on the hundreds of 3x5 inch cards sitting in heaps on the floor!

Such a program could serve as a computerized version of the standard classic of academic writers, **The Manual of Style.** It could also serve the pedagogical purpose of instructing people on how to organize and implement a research project, a skill that is not all that prevalent on most campuses. In addition, as already said, the program can also serve as a tutorial. This article-program package has been written with these possibilities in mind. And the program itself assists the researcher-writer in all of the ways mentioned above, and more.

Anatomical Design of Word-Research Processing Program

Now that I have an idea about the general problem, and the goals and results of the program that I have as an idea in my mind, I must develop a design of the basic components and parts of a program that does what I want it to do.

What are the basic components? What basic functions do I want to appear on the main menu of the operating program?

Notice that even at this stage I am concerned with how my program "screens" will appear. This is an important consideration for what is called structured programming. Structured programming has as one of its main goals the integration of all the phases and parts of a program into a comprehensive view so that things will "hang together" better in a working program. In this particular case, it is good to try to "en-VISION" how an idea will look when it is coded and appears on the monitor of your comupter.

On the screen, my menu will appear as follows:

Operational Choices:
I. Conceptualizing of Research routines
II. Documentation
III. Organizing and Writing-assistance routine

These three are the basic components or modules of the program. I chose these as the components since these are the three basic steps in any research program, and I could thus gather the routines that work for each stage together, work on each one separately, and develop the main organization of the layout of the finished program in my APPLE's memory.

When conceptualizing research projects, certain functions are constants. We must conceive of a topic, plan and organize a project, plan and organize a report, and generally keep these two structures in mind simultaneously.

When documenting research, we need to be able to keep track of authors, books, notes, comments, and the alphabetization of a bibliography.

Finally, when organizing and writing a report, we need to be able to sort out our notes and comments into manageable sections so that a smooth flow of writing plus documentation can be maintained, and we need constantly to be able to check back to previous work, to previously written about notes and comments. These functions are performed in the third module of the program.

Having decided on my main modules, I now need to decide on what functions each will perform. That is, I need the menus for the three modules. These menus will appear on the monitor whenever I select one or another of the main modules when the program is RUNning.

Module I Menu: Conceptualizing & Planning
1. **Built-in model project design**
2. **Assisted development of researcher's own project model**
3. **Built-in model of paper**
4. **Researcher's model of paper**
5. **Model of the entire structure of both project & paper**

Module II Menu: Documentation
1. **Author/title listing**
2. **Author/title search**
3. **Notes**
4. **Notes and comments by book**
5. **Alphabetized bibliography**

Module III Menu: Organizing & Writing
1. **Pre-ordered notes & comments**
2. **Outline + Notes & Comments**
3. **Freeform organizations**

Another aspect of structured programming which is worth mentioning is that the better structured a program is, the more of its aspects there are that interlock and support each other. Since programs have logical, physical, and RUN features, a structured program is one where these three kinds of features support each other. Take a simple example. On RUN, Applesoft BASIC jumps to the earliest line number of the program, and works its way through the program from there. In addition, every time a GOSUB is executed, the APPLE's BASIC jumps back to the earliest line number and scans for the line number of the GOSUB in question. Now, if it takes time to make these scans, wouldn't it make sense to put the most frequently called GOSUBs on the earliest line numbers? And if this were done, wouldn't this be the matching up of the machine's RUN features with the physical layout of the program? Wouldn't it be more economical in time, for the machine to be able to find most called GOSUBs on the earliest lines of the program?

Notice here that if this suggestion is followed, then the format of most programs in BASIC on the APPLE would have to be revised. Instead of putting the Main Program at the early line numbers, and the subroutines at the highest line numbers, it would be better to invert this structure and to place subroutines on early lines and infrequently called Main Programs on the highest line numbers. This is what I have done in the present program.

While I try to make the physical layout of the program match up with the way the machine works, I also try to get these two to match the logical structure of the program. These three different features that match can be visualized by the diagram below:

Features of:

BASIC	**Physical Layout**	**LOGIC**
Line # 1000	Data	Information/Research
Line # 900	Main Program & Menu	Basic steps in Research
Line # 800	Conceptualizing Module	Subprogram #1
Line # 700	Documentation Module	Subprogram #2
Line # 600	Organizing Module	Subprogram #3
Line # 500	Built-in Data	Subprogram #1
Line # 400	Data Read subroutine	Subroutine to Main program
Line # 300	Printing subroutines	General use subroutines
Line # 200	Control subroutines	General subroutines
Line # 100	Menu subroutines	General subroutines
Line #0- 99	Micro subroutines	Most frequently used subroutines

There is yet a fourth thing correlated with these three, and that is the process whereby a research paper is written. The program is so structured that while one may switch at will from one routine or module as frequently as one wishes, nonetheless by following the RUN of the program and choosing each menu selection sequentially, then one will be simultaneously undertaking and following the actual steps that need to be taken from conception to finished paper. This will be taken up further in the next part.

Thus far we have--on paper--the basic outline of the program: we know how the program will look organizationally when it is typed into the APPLE, we know how efficiently that layout is likely to work, and we know the logical structure of the program.

Constructing the Program

Before proceeding to map out the flowcharts of the program, and beginning to code, we need to understand the basic flow of work in researching that the program outlines. Whenever someone has a research project to undertake, there are seven steps or stages in the process. A program intended to help someone along this process should mirror that process in some way. Here are the steps:

1. *An assignment or idea for a project arises in the mind.*
2. *How to formulate the idea in words?*
3. *How to design an outline of the parts of the idea?*
4. *How to construct the report of the project?*
5. *How is the data structured logically and sequentially?*
6. *Gather notes & documentation, and store together.*
7. *Organize research.*

A suitable research-assistance program would be one that was able to formalize, that is, put into an algorithm, the little steps involved in each of these steps, and to find ways of aiding the process along. Now is the time to start coding the program, and developing the routines to carry out all the tasks that will be required.

Coding and keying in the basic outline of the program as depicted in the correlated features diagrammed earlier is an easy task, relatively speaking. I program the way some painters paint--I begin with the broad strokes (lines of code) and work progressively and sequentially on more detailed features. Thus when I begin coding, my first few lines of code will map out the outline of the program:

```
1 REM BY D. KELLY
2 GOTO 900: REM TO MAIN PROGRAM
3: :
4 REM MICRO SUBROUTINE LOCATION
99: :
600 REM MAIN SUBMODULE #1
698 RETURN
699: :
700 REM MODULE #2
798 RETURN
799: :
800 REM MODULE #3
898 RETURN
899: :
900 REM MAIN PROGRAM
998 END
999: :
1000 REM DATA
```

Note that this is already a working program. If I added just one line, 910 GOSUB 600: GOSUB 700: GOSUB 800, the program will run through it's paces, in exactly the way it will do it when all the routines are coded in and the program approaches 1000 lines of code or more. All I will in effect do to this outline, from here on out, is fill in the details. If you key in the above program and RUN it, nothing will appear on your screen except the cursor. So add the following lines at 601,701, and 801: PRINT "XXXXXXX": Get G$. Each GOSUB on line 910 will now pause for you to hit a key before proceeding to the next GOSUB. This is how you proceed in the program coding: you imagine the steps in research, and you devise coded programs to implement them. You also visualize how you want the screen to appear when the program RUNs.

Since words and characters will be appearing on the screen, it will always be necessary to have lines of code that print out lines of various characters, so lines 4-10 can be filled with one-line subroutines that can be called whenever I want them in the program. Since they'll be called frequently, I'll put them on the earliest lines, though I may reserve line 4 for *the* most frequently used subroutine in my programs. So this little block of code may appear as follows:

```
4 PRINT "?-";: GET G$: RETURN
5 FOR A = 1 TO 40: PRINT "%";: NEXT A: RETURN
6 FOR A = 1 TO 40: PRINT "-";: NEXT A RETURN
8 FOR A = 1 TO 40: PRINT " + ";: NEXT A RETURN
9: :
```

The function of line #4 is to give me a way of controlling the process and flow of a RUN, enabling me to make immediate choices. Using the above subroutines, I can improve on my main program, and on a RUN by adding and revising the following lines of code. Key them in now, and try them for yourself.

```
900 REM MAIN PROGRAM
902 HOME: GOSUB 5
904 GOSUB 800: GOSUB 4
906 GOSUB 700: GOSUB 4
908 GOSUB 600: GOSUB 4
910 PRINT "WHAT NOW? 1 = END:2 = REPEAT:"
912 GOSUB 4
914 IF G$ = "1" THEN END
916 IF G$ = "2" THEN GOTO 902
```

At this point, I am ready to get down to the task of keying code for the menu-screens of the three main modules, and to develop the code for the main menu itself. After that is done, I will be ready to turn to the task of developing the code to perform actually useful functions.

You can make the choice of placing your menus in the modules or as calls to subroutines located elsewhere. Thus the main menu can be keyed in

Hard Disks For The APPLE

Philip Castevens

One thing seems obvious about this microcomputer business - it is very much a function of time. Back in April 1980, I purchased what I thought at the time was one of the most advanced micro-systems available: an Apple II Plus, two Disk II drives, and a Paper Tiger (IDS 440) printer.

Well, that was "way back then," and what was "most advanced" then is not necessarily "most advanced" now (four months later). My ego suffers a little when I think of the Apple III with 128K of main memory, an 80-column monitor, twice the speed, a numeric Keypad, and a correspondence-quality IDS 460 printer.

But, I really should not complain. This experience helped me to remember that over-involvement in material concerns such as these ultimately leads to disappointment. And, I have not nearly explored fully the potential of my current system.

And what about hard disks? It is also very difficult to draw a bead on this rapidly moving target. However, since I am involved in developing business systems for the Apple, I have done some research into this situation and would like to share my information with you. Keep in mind that I do not have any personal experience with any of the drives mentioned below and that these comments are my opinions concerning the information I have gathered from magazines, brochures, conversations, etc.. In many ways this is a beginners perspective.

The Past

The following is a brief list of the milestones leading up to the development of hard disks for the Apple:

YEAR	EVENT
1939	Work began on the Mark I (proposed by Howard Aiken), the first operational automatic computer.
1943	The ENIAC (an electronic version of the Mark I) project began.
1959	Transistors began to replace vacuum tubes. The beginning of "second-generation" machines.
1965	"Third-generation" computers (e.g., IBM 360) became available. Featured miniaturized circuits, data communication, etc..
1970	Intel develops the first microprocessor chip.
1975	Steve Jobs and Steve Woznak invent the Apple.
1979	Corvus Systems, Inc. introduces a 10 Megabyte (10M) hard disk for the Apple.

The Present

So here we are in 1980 and there are several hard disk systems for the Apple.

CORVUS 11AP

Corvus is into this thing in a big way. They are doing lots of promotion and offer related systems such as The Mirror (to back the 11AP up on tape) and The Constellation (multiplexer that allows simultaneous access by multiple Apples). Also, there seems to be more software developed by independent sources for use with the 11AP than is available for the other hard disks.

Physically, the Corvus 11AP has the approximate size and shape of a shoe box (but is heavier). It is a high-performance "Winchester" drive and is not removable. It is controlled by a Z-80 processor (which can handle up to 4 disks) and comes with a standard interface card which plugs into slot 6 of the Apple.

Both DOS (for BASIC files) and PASCAL are available. Under the DOS interface, the disk is formatted into multiple images of a standard diskette. This makes it necessary for the program to specify which "volume" to access when dealing with files on the 11AP. Since there can be as many as 82 volumes on-line simultaneously, this could prove awkward unless a file management system such as "Corvus FMS" (see "SOFTWARE" section below) is used. This is not a problem with PASCAL and the drive can be configured as one 10M volume. Cost is $5350 with controller and Apple interface.

LOBO Model 1850

LOBO recently introduced the Model 1850 which "consists of an 8-in. floppy disk drive and an 8-in. fixed Winchester drive housed in the same cabinet. The two drives share the same power supply and disk controller, with the floppy acting as backup medium for the fixed-disk drive." The hard disk can be either 5M or 10M. The floppy drive "is available in a maximum configuration of 1.6M bytes" (COMPUTERWORLD, 6-16-80, pp. 69-70). With prices ranging from $3495 to $4695, this is very attractive.

WIZARD 10

This is a 10M Winchester drive which can be formatted as one file and comes with controller and interface for the Apple. Cost is $4975.

CAMEO DC-500 Cartridge Disk

This is a 10M hard disk system offered by Cameo Data Systems, Inc.. The big advantage here is that it consists of a 5M fixed portion and a 5M removable portion. The removable cartridges can be used for backup purposes, to switch application, and for archival storage. Cost is $5995.

The volume approach to DOS files is similar to Corvus and Cameo is reportedly working on a PASCAL interfacce. They offer an interesting 10-day free trial program.

16K Ram Expansion Board for the Apple II*

$195.00

- switch selection of RAM or mother board ROM language
- includes installation and use instructions
- expands your 48K Apple to 64K of programmable memory
- plugs in slot 0
- allows you to run Apple's new Fortran package and other languages
- eliminates the need for an Applesoft* or Integer Basic ROM Card

Visa & Master Charge accepted
Shipping & handling will be added unless the order is accompanied by a check or money order
NC residents add 4% sales tax
*Apple II and Applesoft are trademarks of Apple Computer, Inc.
Allow 4-6 weeks for delivery

ANDROMEDA
COMPUTER SYSTEMS
P.O. Box 19144
Greensboro, NC 27410
(919) 852-1482

CORVUS Mirror

Earlier this year, Corvus announced The Mirror to solve the backup problem for their hard disks. It works in combination with a video cassette recorder and can store up to 100M on one cassette. Cost is $790 in addition to the cost of the recorder. It takes only 10 minutes for the complete backup process.

CORVUS Constellation

The Constellation multiplexer allows from two to 64 Apples to be linked together and share up to 40M. Cost is $750 plus $235 for each additional interface.

MECA Tape II System

This system includes the hardware and software interface for their Beta-1 Cassette Tape System. It allows random access to 500K bytes per drive. A master and from one to three slave drives are available.

MECA promotes their product as (1) a backup system for disks and (2) a disk replacement. Under $1000.

NESTAR Cluster-One (Model A)

This allows up to 64 Apples to "share data, access the same files and communicate with one another at distances up to 1000 ft." (Computerworld, 2-4-80, p.59) and to use the same peripherals. The Cluster-One comes with either 1.2M 8-in. floppies (double-sided) or 16.5M Winchester-type hard disk subsystem. Prices start at $6000.

Software

There are at least two companies offering access methods for the Corvus 11AP. Alpine Software has developed "Corvus FMS" ($395) which is like an advanced DOS that includes an indexed sequential access method (ISAM) for communicating with large Applesoft files. It also minimizes unused space within files. United Software of America sells "KRAM" (keyed random access method) which works with integer BASIC files and costs $99. Peripherals Unlimited is developing software for the Cameo DC-500 system.

Many products, such as Datacopes Single Disk Sort ($50) will work with hard disks like the Corvus, but only one volume at a time. In fact, it seems that, in general, sorting large files on one of these hard disks is likely to take a long time. This is because of the speed and main memory limitations of the Apple II. The Apple III should work better in this respect.

Note that software is not as sensitive to the passage of time as is hardware. The same software -especially ANS COBOL and UCSD Pascal programs - can be used within many different hardware configurations with minimal conversion.

The Future

Shugart Technology has recently announced a 5.25-in. 6.3M Winchester hard disk drive which sells for under $1000 each in large quantities. Perhaps this unit will become available for use with Apple processors.

It has been rumored that Apple Computer itself may come out with a hard disk of some sort (perhaps as a component along with the Apple III in some business-oriented system).

Venture Development Corp. predicts the rate of shipments of "low-cost, low-performance ... Winchester drives ... will reach 375,000 units in 1984" and that "shipments of high-performance 8-in. fixed drives will reach a level of 54,000 units, while 8-in. cartridge drives will be selling at twice that level." (COMPUTERWORLD, 2-18-80, p. 66)

Personally, I think that a pair of small, low-cost 5M cartridge drives would be a hot-selling addition of Apple systems.

Summary

I do not want to over-emphasize either the products currently available or those which seem likely to become available in the future. For someone considering the purchase of hard disks for the Apple, the decisions should be based on past, present, and future considerations. A balanced approach such as this will probably yield the most satisfaction.

Addresses

ALPINE SOFTWARE
4874 Ridenour
Colorado Springs, CO 80916

APPLE COMPUTER INC.
10260 Bandley Dr.
Cupertino, CA 95014

CAMEO DATA SYSTEMS INC.
1626 Clementine
Anaheim, CA 92802

CORVUS SYSTEMS INC.
2029 O'Toole Ave.
San Jose, CA 95131

DATACOPE-PO DRAWER AA
Hillcrest Station
Little Rock, AR 72205

LOBO DRIVES INT.
935 Camino Del Sur
Goleta, CA 93017

MECA
7026 O.W.S. Road
Yucca Valley, CA 92284

NESTAR SYSTEMS INC.
430 Sherman Ave.
Palo Alto, CA 94306

PERIPHERALS UNLIMITED
6012 Warwood Road
Lakewood, CA 90713

SHUGART TECHNOLOGY
340 El Pueblo Road
Scotts Valley, CA 95066

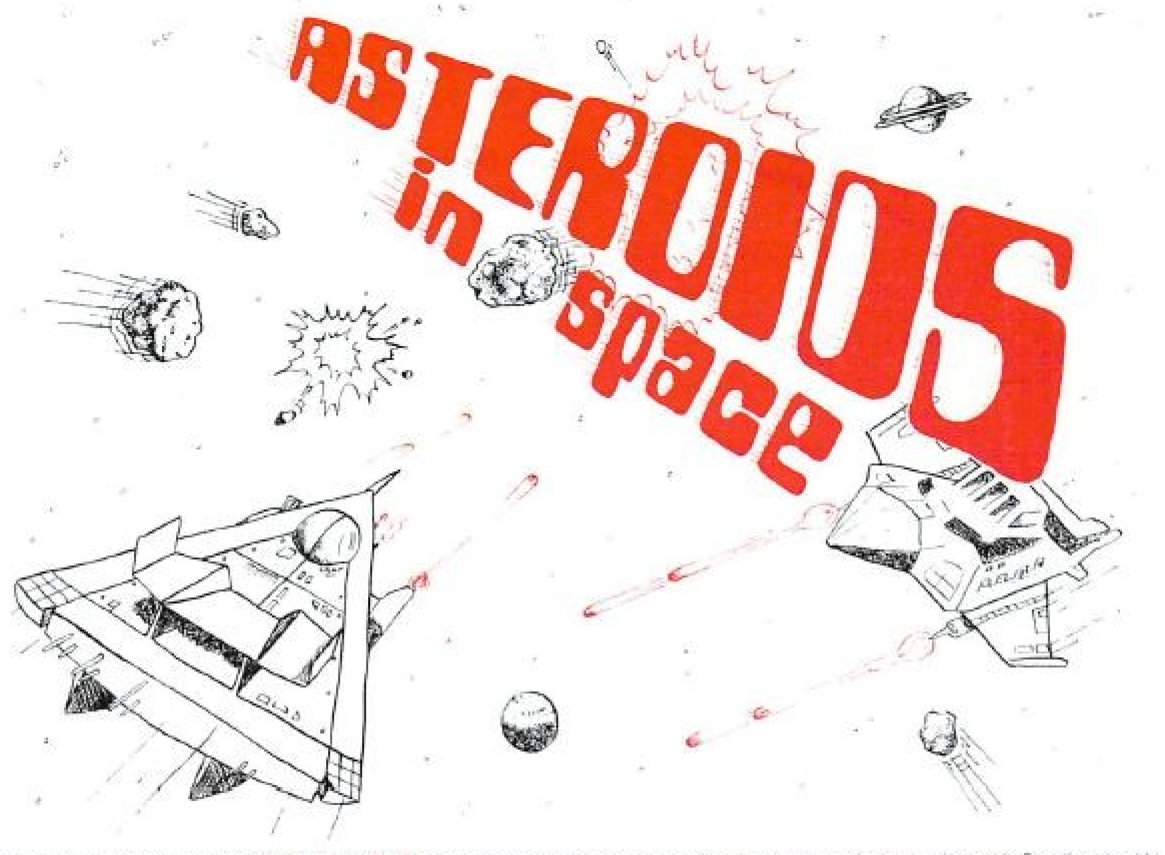
UNITED SOFTWARE OF AMER.
750 Third Avenue
New York, NY 10017

VENTURE DEVELOPMENT CORP
One Washington Street
Wellesley, MA 02181

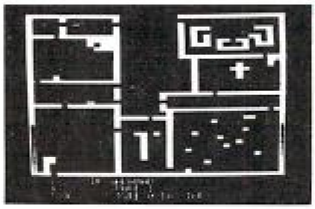
WIZARD
COMPUTER DISTRIBUTORS
P.O. Box 9194
Austin, TX 78766

©

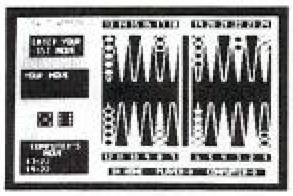
Exciting, entertaining software for the Apple II and Apple II Plus*



If you liked "Invaders", you'll love ASTEROIDS IN SPACE by Bruce Wallace. Your space ship is traveling in the middle of a shower of asteroids. Blast the asteroids with lasers, but beware — big asteroids fragment into small asteroids! The Apple game paddles allow you to rotate your space ship, fire its laser gun, and give it thrust to propel it through endless space. From time to time you will encounter an alien space ship whose mission is to destroy you, so you'd better destroy it first! High resolution graphics and sound effects add to the arcade-like excitement that this program generates. Runs on any Apple II with at least 32K and one disk drive.
On diskette — **$19.95**



FRACAS™ by Stuart Smith. A fantastic adventure game like no other — up to eight players can participate in FRACAS at the same time. Journey in the land of FAROPH, searching for hidden treasure while warding off all sorts of unfriendly and dangerous creatures like the Ten Foot Spider and the Headless Horseman. You and your friends can compete with each other or you can join forces and gang up on the monsters. Your location is presented graphically and sound effects enliven the battles. Save your adventure on diskette or cassette and continue it at some other time. Requires at least 32K of RAM. Cassette: **$19.95** Diskette: **$24.95**



BATTLESHIP COMMANDER™ by Erik Kilk and Matthew Jew. A game of strategy. You and the computer each start out by positioning five ships of different sizes on a ten by ten grid. Then the shooting starts. Place your volleys skillfully — a combination of logic and luck are required to beat the computer. Cartoons show the ships sinking and announce the winner. Sound effects and flashing lights also add to the enjoyment of the game. Requires at least 32K of RAM.
Cassette: **$14.95** Diskette: **$19.95**

FASTGAMMON™ by Bob Christiansen. Sound, hi res color, and cartoons have helped maked this the most popular backgammon-playing game for the Apple II. But don't let these entertaining features fool you — FASTGAMMON plays serious backgammon. Requires at least 24K of RAM.
Cassette: **$19.95** Diskette: **$24.95**

QS QUALITY SOFTWARE

6660 Reseda Blvd., Suite 105, Reseda, CA 91335

WHERE TO GET IT: Call us at (213) 344-6599 for the name of the Quality Software dealer nearest you. If necessary, you may order directly from us. MasterCharge and Visa cardholders may place orders by telephone and we will deduct $1 from orders over $19 to compensate for phone charges. Or mail your order to the address above. California residents add 6% sales tax. SHIPPING CHARGES: Within North America orders must include $1.50 for first class shipping and handling. Outside North America the charge for airmail shipping and handling is $5.00 — payable in U.S. currency.

SEND FOR OUR FREE CATALOG

COMPUMART LETS YOU PUT APPLE ON TRIAL

WITH OUR EXCLUSIVE 10-DAY FREE RETURN ON ALL THESE GREAT APPLE COMPUTERS AND ACCESSORIES.

CompuMart carries complete lines of microcomputers from Apple to Zenith, as well as thousands of microcomputer peripherals and accessories. Write for our FREE 36 page catalog.

Apple Computer

We carry the most complete inventory of Apple computers, peripherals, and software. CALL!

Our Christmas Apple Special: Save over $250 on our most popular Apple System. System includes a 48K Apple II, Apple Disk & Controller, and a Sup R Mod RF Modulator.

List: **$2,020**

Compumart Sale Price: **$1,769**

New from Apple for the Apple II:

DOS 3.3 Convert disks to 16 sector format for 23% more storage and faster access	$60
Apple Plot The perfect graphic complement for Visicalc	$70
Dow Jones News & Quotes	$95
Adventure (Uses 48K)	$35
DOS Tool Kit	$75
Apple Fortran	$200
Silentype Printer w/Xface	$595
Visicalc	$149
Tax Planner	$120

From Symtech & Info Unlimited

Super Sound Generator (mono)	$159
(stereo)	$259
Light Pen	$249
X-10 Controller (plugs into paddle port)	$49
Apple Sync Controller	$49

From Personal Software

Visicalc	$149
Desk top plan	$99

New from Videx! — Video Term 80 Col. x 24 line 7 x 9 matrix, plug in compatible board for the Apple II. Price .... $325 without graphics EPROM. With graphics EPROM .... $350.

New from MUSE

The Voice	$39.95
Super Text	$99.00
Address Book	$49.95

Mountain Hardware — Expansion accessories for your Apple

Introl/X-10 System	$289
Super Talker	$299
The Music System	$545
ROM plus board w/keyboard filter	$199
Clock Calendar	$280
16 Channel A to D Converter	$350
Apple Expansion Chassis	$650
ROM Writer	$175

Miscellaneous Apple II Accessories:

Easy Writer (80 col. need Videx)	$249
Easy Mover	$49
Easy Mailer	$69
Dysan Diskettes	ea. $5
S.S.M. Serial & Parallel Apple Interface	$225
ABT's Numeric Key Pad	$110

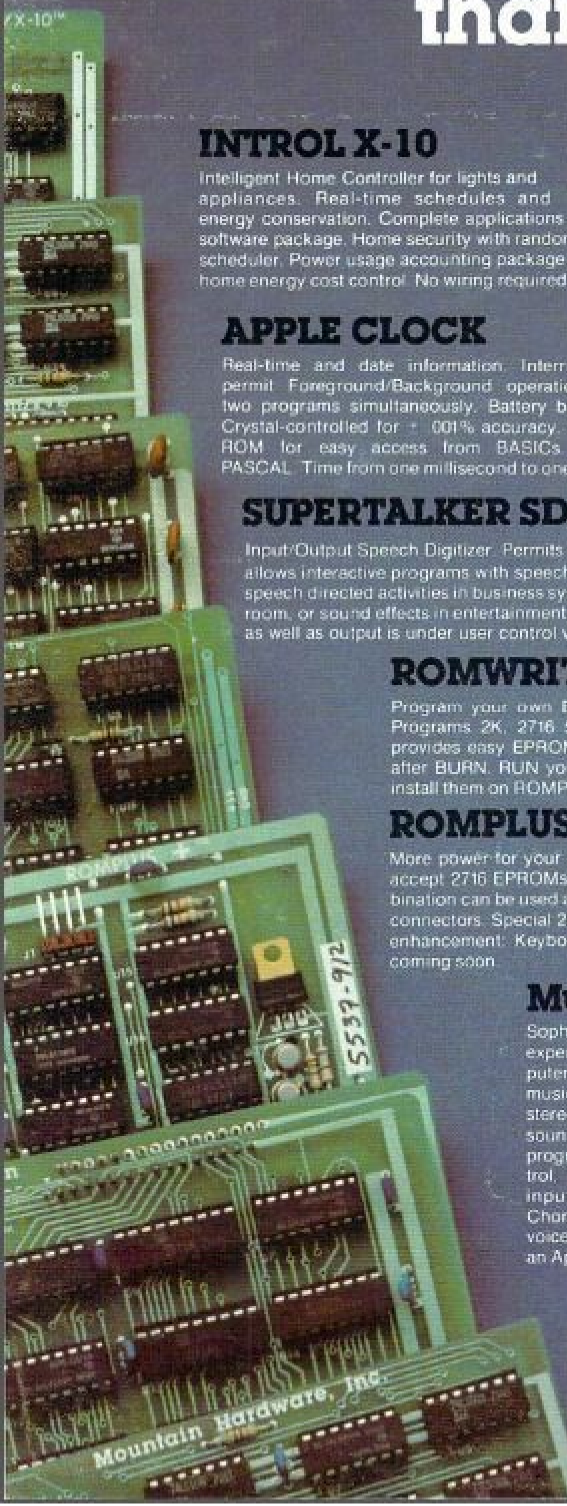
270 THIRD ST. DEPT 30 **COMPUMART** CAMBRIDGE, MA 02142

TOLL FREE 1-800-343-5504 **IN MASS 617-491-2700**

apple computer
Authorized Dealer

Mountain Computer makes more peripherals for the Apple Computer than Anybody.

and . . . a place to put them



INTROL X-10

Intelligent Home Controller for lights and appliances. Real-time schedules and energy conservation. Complete applications software package. Home security with random scheduler. Power usage accounting package for home energy cost control. No wiring required.

APPLE CLOCK

Real-time and date information. Interrupts permit Foreground/Background operation of two programs simultaneously. Battery back-up. Crystal-controlled for ± .001% accuracy. Onboard ROM for easy access from BASICs. Supports PASCAL. Time from one millisecond to one year.

SUPERTALKER SD200

Input/Output Speech Digitizer. Permits talking programs. I/O capability allows interactive programs with speech-prompted inputs. Use output for speech directed activities in business systems, announcements in a control-room, or sound effects in entertainment programs. Easy to use because input as well as output is under user control with special software operating system.

ROMWRITER

Program your own EPROMs. Create your own firmware. Programs 2K, 2716 5V EPROMs. Disk software package provides easy EPROM programming. EPROMs are verified after BURN. RUN your programs from on-board socket or install them on ROMPLUS+.

ROMPLUS+

More power for your system through firmware. Six sockets accept 2716 EPROMs or ROM equivalents. Six or any combination can be used at once. Scratch-pad RAM and two TTL connectors. Special 2K ROMs available for powerful system enhancement: Keyboard Filter ROM—COPYROM—Others coming soon.

MusicSystem

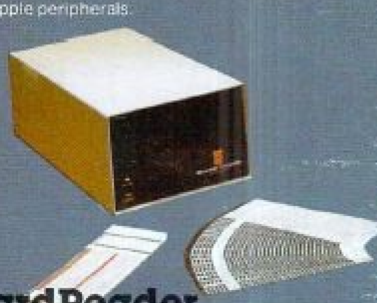
Sophistication previously available only on experimental mini and mainframe computer synthesizers. Digital instrumental music synthesizer system. 16 voices in stereo. Instrument definitions simulate the sound of real instruments—and more. Fully programmable waveforms. Envelope Control. Composition system—sheet music input using standard music notation. Chords and multi-part scoring up to 16 voices. A true instrument that anyone with an Apple can play.

A/D+D/A

16 channels analog to digital input. 16 channels digital to analog output. Eight bit resolution. Super-fast 9μ sec. conversion time. Monitor and output to the real world. All on one card.

EXPANSION CHASSIS

By popular demand! Eight more slots for your Apple. Attractive sturdy enclosure. Its own heavy duty power supply. Easy to use. Address cards in Expansion Chassis the same way as in your Apple. Only one additional command to specify in Apple or in Expansion Chassis. Compatible with all Apple peripherals.



CardReader

At last! An intelligent, high-quality device for data entry from user-marked cards. Implement BASIC programming, examination scoring, inventory maintenance and other applications requiring off-line data preparation for batch entry later. Connects to any computer having RS-232 interface. Software and cards are available for jobs in business, science and education.

MOUNTAIN COMPUTER has the most comprehensive line of Apple peripherals available. Anywere. From anybody. We know the Apple inside and out and are committed to providing the most innovative and unique products to expand and enhance its capabilities and use. After all, we were the first company to make an Apple peripheral—except Apple Computer.

Available at Apple Dealers worldwide.

Mountain Computer
INCORPORATED

300 Harvey West Blvd., Santa Cruz, CA 95060
(408) 429-8600 TWX 910 598-4504

*Based on Manufacturer's published catalogs—Apr. 1980

Apple is a trademark of Apple Computer Inc.

NOW THE SOFTCARD™ CAN TAKE YOU BEYOND THE BASICS.

You probably know about the SoftCard — our ingenious circuit card that converts an Apple II® into a Z-80® machine running CP/M®.

You may even know that with the SoftCard, you get Microsoft's powerful BASIC — extended to support Apple graphics and many other features.

Now, whenever you're ready to get beyond the BASICs, the SoftCard can take you into whole new realms. Starting with two advanced language packages from Microsoft.

FORTRAN AND COBOL TO GO.

Now you can run the world's most popular engineering/scientific language and the most popular business language on your Apple. Think what that means: you can choose from literally thousands of "off-the-shelf" applications programs, and have them working with little conversion. Or design your own programs, taking advantage of all the problem-solving power these specialized languages give you.

FORTRAN-80

A complete ANSI-standard FORTRAN (except COMPLEX type), with important enhancements. The extremely fast compiler performs extensive code optimization, and, since it doesn't require a "P-code" interpreter at run time, your programs will typically execute 2-3 times faster than with Apple FORTRAN.

FORTRAN is easy to learn if you know BASIC, and the package includes a huge library of floating point, math, and I/O routines you can use in all your programs.

COBOL-80

Virtually the only choice for serious business data processing. It's ANSI 1974 standard COBOL, with many user-oriented features added: formatted screen support for CRT terminals, simple segmenting of very large programs, powerful file handling capability, trace debugging, and much more. A separate Sort package is coming soon.

FORTRAN-80 and COBOL-80 are just two more reasons why the Apple with SoftCard is the world's most versatile personal computer. Get all the exciting details from your Microsoft dealer today. And start getting beyond the BASICs.

MICROSOFT Consumer Products, 400 108th Ave. N.E., Suite 200, Bellevue, WA 98004. (206) 454-1315.

MICROSOFT

Load PET Program Tapes Into The APPLE II

Keith Falkner

Applesoft BASIC and COMMODORE BASIC are very similar languages. For this reason it is feasible to load PET programs into an APPLE II running Applesoft BASIC in either ROM or RAM. There are problems every step of the way, but this program solves most of them, with the result that the APPLE can actually run very many of the fine programs available for the PET. Of course there are some profound differences between the two computers, and programs which exploit features of the PET which are absent from the APPLE may be impossible to run.

I call this program "PET Loader", and it requires an APPLE II with Applesoft BASIC. There are two versions, one for the APPLE II PLUS as shown in the listing, and one for an APPLE II with Applesoft in RAM locations $0800-$3000. To load a program which just filled an 8K PET, an APPLE II PLUS uses 10K, and an APPLE II uses 20K. The PET program can then be saved and loaded exactly as an Applesoft program. PET Loader is not needed again for that program.

The operating instructions for PET Loader are in Table 1. If it detects an error, it prints a single digit to identify the error, followed by the message "ERR" and a beep, then it quits. Table 2 shows what has caused the error, and what to do then. Because the PET writes tape very dependably, and because there are two copies of the program on the tape, poor reads are rare, and the usual result is a cheerful 'OK', just after the traditional beep signalling 'stop the tape'. PET Loader then returns to Applesoft, and the "]" prompt appears.

Now the fun begins, or so the saying goes. There are many differences between the PET and the APPLE, which the program could not resolve. When real intelligence is needed, it all depends on you! A PET program loaded into an APPLE is still not an APPLE program. Here are the major differences which you will have to consider, when you try to produce a usable APPLE program:

1. No Equivalent Verb in the APPLE.
OPEN, CLOSE, VERIFY, and CMD are commands in COMMODORE BASIC. Each of these is translated "STOP", and you will need to decipher the programer's intent and program the equivalent for the APPLE. Refer to COMMODORE's excellent manuals for descriptions of these commands.

2. Specific Device Reference.
Programs containing OPEN and CLOSE will also contain either PRINT# or INPUT#, which are simply translated to PR# and IN# respectively, and will require substantial rework. The devices, by number, are conventionally these:

#0	the keyboard
#1	first cassette drive
#2	second cassette drive
#3	the screen
#4	the printer (or maybe not)
#8	the dual disk drive

3. Reference to Actual Memory Locations.
PEEK, POKE, CALL (SYS in the PET), WAIT, and USR refer to specific locations in memory, and you will need more help than I can offer here.

4. Keys to Move the Cursor.
The PET has ten keys to control the position or action of the cursor. Eight of these are missing from the APPLE. PET Loader translates these as follows:
Two functional equivalents-

CURSOR-LEFT becomes BACKSPACE, and appears as
"GR" in the program.

CURSOR-DOWN becomes LINE-FEED, and appears as
"PR#" in the program.

(Odd as they appear, these actually move the cursor exactly as stated.)

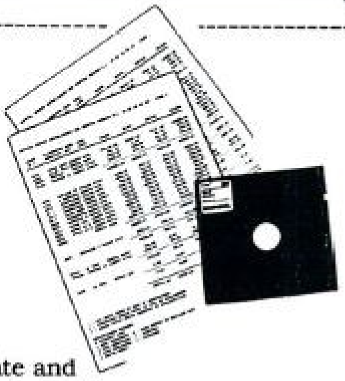
One destructive approximation:
CURSOR-RIGHT becomes SPACE, which obliterates what it should space past. This appears as "COLOR =" in the program.

Seven non-functional comments:
When the program is listed, these are visible, looking like genuine verbs, and it looks as if the name of the key will be printed. For example,

```
100 PRINT "CLEAR"
110 INPUT "INVERSE INSTRUCTIONS NORMAL"
;Z$
```

FOR: apple* **FROM: innerface BUSINESS SYSTEMS**

Capital Asset Managment System™



SIMPLE,
CONCISE,
FUNCTIONAL,
ACCURATE,
EFFICIENT,
USABLE
SOFTWARE from I.B.S.

Having developed accurate and understandable software for both government and business, I.B.S. has at last Gone Public, with C.A.M.S. —

CAPITAL ASSETS MANAGEMENT SYSTEM, a simple, easy-to-use system designed for the APPLE Computer* and you.

- DEPRECIATES assets according to one of the eight approved schedules
- INVESTMENT CREDIT and RECAPTURE are computed automatically
- PROJECTIONS aid in the selection of method and term for new assets
- PERSONALIZED REPORTS in either 80 or 132 column formats
- AUTOMATIC SUBTOTALING selected by the user
- SAFTEY ROUTINES check input for most errors
- AUTOMATIC DISK FILING with user control

CAMS:

With just a few clear and simple keystrokes, you can view any of two hundred assets depreciated FROM any time TO any time, using any of the eight schedules.

By SEEING the results, you can make accurate decisions NOW, when it counts.

CAMS will automatically search your records for assets which qualify for either Investment Credit or Investment Credit Tax Recapture.

CAMS prompts you with clear and simple messages on the screen for all entries. Special safety routines check all practical inputs for errors. Because CAMS maintains all files automatically, it serves not only as a tool for projections, but as a RECORD FILING system as well.

And since no one should be limited by software, INNERFACE has made CAMS for user back-up copies as well. Because of this, you can record an UNLIMITED number of assets on individual disks for privacy and safe-keeping.

CAMS records 23 pieces of information on each asset: GL accounts; memos; dates; costs; method & life; full description; user ID/dept.; other deductions.
CAMS reports on depreciation: Straight-line; 125%, 150%, 200% w/wo automatic switch to Straight-line, Sum-of-year-digits AND Investment Tax Credit & Recapture amounts for each year.
CAMS requires a 48K Apple with disk and (at least) 80-column printer.
Cams is fully supported by professional documentation and by I.B.S.

* APPLE is a trademark of Apple Computer Inc.

innerface BUSINESS SYSTEMS

CAMS is 99.50 $2.00 SHIPPING CA RES ADD 5.97 TAX
box 834, pacific grove CA 93950

In fact, line 100 will neither clear the screen nor print "CLEAR". It will merely print an equal-sign (=). Line 110 will print "INSTRUCTIONS", and no trace will be seen of the INVERSE and NORMAL commands shown in the listing. This behavior can be perplexing, because usually with Applesoft, what you see is what you get. The purpose of these translations is to disclose the intent of the program, so the cursor-keys of the PET are translated thus:

```
PET key: RUV OFF HOME CLR UP DEL INST
Lists as: INVERSE NORMAL HOME CLEAR VLIN
    DEL IN#
Result: nil nil nil " = " nil nil nil
```

... so when you list the program and discover:

```
300 INPUT "HOME PLAY INVERSE
AGAIN NORMAL" ; X$
```

substitute the equivalent APPLE code, which in this case is:

```
300 VTAB 1: PRINT "PLAY";: INVERSE:
PRINT "AGAIN";: NORMAL: INPUT "?";X$
```

For all those keys except INST and DEL, the equivalent in Applesoft is easy to devise, but logic to produce the function of PET's INSERT and DELETE keys is extremely difficult, and APPLE's convoluted screen addressing makes this task truly hairy. Fortunately very few PET programs print INSERT or DELETE characters.

5. Printing of numbers is slightly different.

```
290 X = 4 : Y = - 6
300 PRINT "X IS" ; X ; "Y IS" ; Y ; "."
```

The PET prints: X IS 4 Y IS-6.
THE APPLE PRINTS: X IS4Y IS-6.

The PET prints a blank before positive numbers, and a CURSOR-RIGHT after all numbers; the APPLE does neither. By the way, all four semicolons (;) in line 300 above are optional in both computers.

6. A side effect of TAB.

A PET can TAB over data already on the screen; the APPLE wipes it out, so use an HTAB verb in place of a TAB phrase if this difference mars the display.

```
PET: 40 PRINT TAB(11) "XYZ"
APPLE: 40 HTAB 12 : PRINT "XYZ"
```

7. Computations in Boolean Arithmetic.

In the following lines,

```
400 X = 11 : Y = 6 : Z = X > Y
410 PRINT Z : IF Z THEN 500
```

The PET will set Z to -1, and line 410 will print this result, then go to 500. The APPLE will set Z to +1, and print this different result, then go to 500. In the above example, the difference may not be crucial, but it often can be. Because the PET does bit-by-bit evaluation of the operators OR and AND, in

```
700 X = 11 : Y = 6 : Z = X AND Y
```

PET's result will be Z = 2, because the bit pattern of 11 is 0000 1011 and the bit pattern of 6 is 0000 0110, and these two patterns, ANDed, give 0000 0010, arithmetically 2. APPLE, on the other hand, merely sees that neither X nor Y is FALSE (zero), calls the result TRUE, and gets Z equal to 1. This can be a very subtle pitfall.

8. Random numbers.

RND (0) gives a genuine random number each time in the PET, but in the APPLE, it repeats the previous random number. Simply replace the 0 with a 1.

9. The GET command.

In the PET, GET does not wait for a key to be struck, so the sequence

```
333 GET P$ : IF P$ = "" THEN 333
```

is the customary way for a PET program to wait for a key. Oddly enough, this is completely appropriate in the APPLE, because if the key struck is CTRL-@, then P$ will be the null string. Ignorance of this is an obscure bug in some Applesoft programs. When the PET program is testing for a key but not waiting when no key has been struck, a different approach is needed. For example,

```
PET: 60 GET A$ : IF A$ = "" THEN 100
    APPLE: 60 ON PEEK (-16384) < 128 GOTO 100 :
    GET A$ : IF A$ = "" THEN 60
```

10. Graphics Characters and Lower Case.

The PET can produce lower-case letters and many graphic symbols which the APPLE cannot, and there are two display-modes in the PET. $C1 might mean "a" (old PET), "A" (new PET), or the symbol for the Spade suit (any PET). PET Loader looks for $CF, probably a lower-case "o", in the PET program. If $CF is found, all letters are translated to upper case; if not, graphic symbols are translated into a similar character the APPLE can produce.

11. Direct Screen Addressing.

In both computers, the video screen occupies a part of memory, and a POKE to a storage location in the screen memory will produce a character on the screen. The relationship between memory location and screen position in the PET is straight-forward, but it is quite complicated in the APPLE, and therefore not often used. Nevertheless, it is worth mastering, because there are hordes of PET programs which use this technique, and a lot of them are attractive games. The PET has 25 lines of 40 columns, and the memory location of each byte of the screen can be computed thus (the expression is not written in BASIC):

```
LOCATION = 32768 + 40 * LINE +
COLUMN
```

where the upper left corner is LINE 0, COLUMN 0.

The APPLE has 24 lines of 40 columns, and the memory location of each byte can be calculated by:

```
LOCATION = XL% (LINE) + COLUMN
```

where the array XL% has been initialized thus:

```
1000 DIM XL% (23) : FOR I © 0 TO 7 :
        XL% (I) = 1024 + 128 * I :
        XL% (I + 8) = 1064 + 128 * I :
        XL% (I +16) = 1104 + 128 * I : NEXT
```

As before, the upper left corner is LINE 0, COL-

UMN 0. In applying this tactic take care not to let COLUMN exceed 39, or you will cause destruction of some important values in memory. For example, a POKE to valid LINE 23, and invalid COLUMN 49, will likely cause loss of your BASIC program.

12. Numeric Keypad.
Programs which use screen-POKEs to move pieces of a game around the screen use the keys 1-9 to indicate the direction of motion. This is satisfactory on the PET, because these keys form a square, with 7,8,9 above 4,5,6 above 1,2,3 and it is natural that, if the 5-key means 'stop', the 8-key means 'up', and the 3-key means 'down and right'. APPLE'S numeric keys do not form a square, so some substitute must be devised. None is immediately obvious, but perhaps the parallelogram formed by R,T,Y above F,G,H above V,B,N would serve, since the 'BELL' on the G-key can be easily remembered as being a 'home' position. Often the game can be improved by substituting use of the game paddles for these keys; why should you accept the limitations of the PET?

13. Sound.
Without an accessory, the PET is silent, but a simple way of making noises has become very popular. Discovery of a POKE to 59466 is strong evidence that this technique is in use. Here is how to deduce the programmer's intent:

POKE 59467,16	**startmaking noises (musical tones)**
POKE 59467,0	**stop making noise altogether**
POKE 59466,15 (or 51 or 85)	**typical values to define an octave**
POKE 59464,X	**high X is a low note, and vice versa.**

It is important to remember that once the PET issues three POKEs, the sound is continuous, whereas the APPLE must continually address the speaker to keep making noise. This part of the conversion can be left for last, because the POKEs address APPLE's Read-Only Memory, which is unaffected by POKE.

14. Real-time clock.
The PET has a genuine timer, which programs can address in two ways. Variable TI increases by 60 every second, and can be read but not written; string TI$ is six numeric characters, in the format HHMMSS, and can be read and written.

```
400 PRINT "THE TIME IS" ; TI$
```

TI$ is computed from the instantaneous value of TI, and formatted as up to six digits. If you try running a PET program in your APPLE, and it just stalls, doing nothing at all, press CTRL-C to stop it, and you may find lines like

```
700 X = TI + 60
710 IF TI < X THEN 710
```

Line 710 is merely waiting for a second to elapse, and in the timeless APPLE, it never will. Substitute a FOR-NEXT loop of the appropriate duration.

15. PI.
A single key on the PET provides the number PI, 3.14159265. No such facility exists in the APPLE, so PI is translated into a character which prints as "UNDEF'D FUNCTION". This causes "?SYNTAX ERROR", and is simple to correct.

16. The INPUT statement has subtle differences.
The PET provides a question mark after the prompt for INPUT, and for neatness, you should supply one. If the PET issues an INPUT statement and the user merely presses RETURN, execution of the program ceases at once. This is such a nuisance that programs with any elegance guard against it in a variety of ways, for example:

```
50 INPUT "WHAT NOW > > + < < <" ; X$
```

where '>' and '<' stand for keys to move the cursor right and left. You can tidy this up as you supply the question mark for the prompt.

As you ponder all the differences and incompatibilities listed, and the other differences which you will find, you may wonder if the resulting program will be worth the work. In fact, PET Loader is a potent and reliable tool, and you will appreciate it. The tape-reading process has proven particularly sound, and has error-recovery procedures even better than those in the PET.

I received a lot of help in this project, all of it from Gordon Campbell of Toronto, who provided manuals, clippings, advice, and many PET pro-

NEW FOR APPLE 2 & APPLE 2 Plus

ASTEROID

A REAL TIME HI-RESOLUTION GRAPHICS GAME WITH SUPER SOUND EFFECTS. THREE SKILL LEVELS - NOVICE, AVERAGE ("Arcade" difficulty") and EXPERT WHERE ASTEROIDS ARE ACTUALLY ATTRACTED TO YOUR SHIP!

ASTEROID

Now Available

32K TAPE - $14.95
32K DISK - $19.95

From Your Local Computer Store Or Send Order to:

© Copyright 1980

Adventure INTERNATIONAL
A DIVISION OF SCOTT ADAMS, INC

BOX 3435
LONGWOOD FLA 32750
(305) 862-6917

DEALER INQUIRIES INVITED

grams. He received some Applesoft programs in exchange, and he loaded them into his PET! But that is his story, and you can read it in the Sep/Oct 1980 issue of COMPUTE.

To get the source and object of both versions of PET Loader, as well as two accessory programs I have developed, send me ten dollars, and I will mail them to you on a cassette, together with printed copies of Table 1 and Table 2.

TABLE 1.
OPERATING INSTRUCTIONS

1. Before loading PET Loader, ensure the BASIC pointers in page zero are normal, by either issuing the DOS command "FP", or powering off and on.
2. Load PET Loader from tape or disk and run it. It will ask you to "PLAY" a PET program tape and press a key. Do so (or press CTRL-C to quit).
3. Any clicks you hear indicate unreadable bytes on the tape. Before or between programs on the tape, these are perfectly normal events.
4. After 10-40 seconds, the name of a program will appear near the top of the screen. If you wish to bypass this program, press CTRL-X promptly, and the name will disappear, then you can press a key to try the next program.
5. Many error conditions are tested at this point. Refer to Table 2 if any of them occurs. The signal is a digit, the letters "ERR", and a "BEEP".
6. The image of the program will be read into memory, and simultaneously be displayed in a 32-character window. Some of it will appear to make sense.
7. Relax. PET programs load slowly, for example 20K in nine minutes. If the speaker clicks during the load, the wait will be doubled, because PET Loader will have to read the second copy of the program as well.
8. When the whole program is loaded, a "BEEP" is issued. Stop the tape.
9. There is a very brief pause while tokens and data are translated, then the message "OK" is issued, and the Applesoft prompt "]" reappears.
10. You now have a PET program in your APPLE. Refer to the article, to turn it into an APPLE program (there's always a catch, isn't there?).
11. To rerun PET Loader, just type "&". You don't need to reload the program.

MAKING IT WORK

The best way to enter PET Loader into your APPLE is to have an Assembler. I used Programma's mighty ASM/65, whose only idiosyncracy is the use of "<" and ">" to denote the low and high halves of a two-byte address. (The ".FILE" lines serve to connect pieces of a program which will not all fit in memory.) Next best is the Mini-Assembler which is part of Integer BASIC. Worst is byte-by-byte keying through the Machine-Language Monitor. If you are capable of doing any of these, you already know how (this is not the program to learn the technique on!).

To package PET Loader as an Applesoft program, execute these instructions after

TABLE 2.
ERRORS AND RECOVERY

If PET Loader detects an error it cannot handle, it issues an error message and quits, returning control to Applesoft. The error message is a digit and "ERR".

MESSAGE	MEANING AND RECOVERY
1 ERR	A long search for a Header Label has not found one. If this really is a PET tape, perhaps the volume is improperly set. PET tapes are louder than APPLE tapes, so adjust the volume by ear and try again.
2 ERR	Neither copy of the label could be read correctly. This might also be improper volume, but is probably a bedraggled tape. Either try again, or acquire another copy of the program.
3 ERR	The header label was read correctly, and a code in it says that the contents of this file is not a BASIC program, but data. PET Loader reads programs only. Try another file or another tape.
4 ERR	This program is too big to fit in memory. Perhaps the pointer at location 115-116 ($73-74) was altered by HIMEN:, so issue the DOS command "FP", or HIMEN:16384 (or what have you), and try again.
5 ERR	After the header label was read correctly, the program should follow. It was not found, perhaps because the PET user aborted the "SAVE". If repeated tries cause the same message, that is most likely.
6 ERR	This program loads at an unusual address in the PET. Possibilities are: proprietary program, machine-language or partly so, or some attempt to make the program difficult to copy (or use?). Forget it!

To try again, just type "&" and press RETURN. With RAM Applesoft "CALL 1013". Use this tactic even if you have typed "NEW" or "FP", or have pressed RESET!

loading it into locations $800-$BF6, using the Machine-Language Monitor:

```
*67:01 08 00 0C 00 0C 00 0C
*AF:00 0C N E003G
(which gets you back to Applesoft)
```

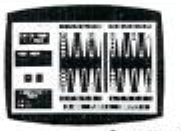
then LIST the program; you should get only one line: 65535 CALL 2064 : END PET Loader can now be saved, loaded, and run exactly as an Applesoft program, except that PET Loader cannot be saved after it has been run. The program occupies locations 2048-3071 ($800-$BFF), and makes them unavailable to Applesoft. This costs 1024 bytes of memory, but saves having to reload PET Loader to convert the next program. To restore the pointers to normal, issue the DOS command "FP", or type "CALL -151" then enter Applesoft "cold" with CTRL-B.

The version of PET Loader which operates with RAM Applesoft occupies memory locations

Software for the Apple II and Apple II Plus*

ASTEROIDS IN SPACE™
By Bruce Wallace
An exciting space action game! Your space ship is traveling in the middle of a shower of asteroids. Blast the asteroids with lasers, but beware — big asteroids fragment into small asteroids! The Apple game paddles allow you to rotate your space ship, fire its laser gun, and give it thrust to propel it through endless space. From time to time you will encounter an alien space ship whose mission is to destroy you, so you'd better destroy it first! High resolution graphics and sound effects add to the arcade like excitement that this program generates. Runs on any Apple II with at least 32K of RAM and one disk drive.

On Diskette - $19.95



FASTGAMMON™ By Bob Christiansen.
Sound, hi res, color, and musical cartoons have helped make this the most popular backgammon playing game for the Apple II. But don't let these entertaining features fool you — FASTGAMMON plays serious backgammon. Runs on any Apple II with at least 24K of RAM.

Cassette - $19.95 Diskette - $24.95

ASTROAPPLE™ by Bob Male.
Your Apple computer becomes your astrologer, generating horoscopes and forecasts based on the computed positions of the heavenly bodies. This program offers a delightful and stimulating way to entertain friends. ASTROAPPLE produces natal horoscopes (birth charts) for each person based on his or her birth data. Any two people may be compared for physical, emotional, and intellectual compatibility. The program is written in Applesoft BASIC with machine language subroutines. It requires either RAM or ROM Applesoft and at least 32K of memory.

Cassette - $14.95 Diskette - $19.95

FRACAS™ by Stuart Smith.
A fantastic adventure game like no other! Up to eight players can participate in FRACAS at the same time. Journey in the land of FAROPH, searching for hidden treasure while warding off all sorts of unfriendly and dangerous creatures. You and your friends can compete with each other or you can join forces and gang up on the monsters. Your location is presented graphically and sound effects enliven the battles. Save your adventure on diskette or cassette and continue it at some other time. Both integer BASIC and Applesoft versions included. Requires at least 32K of RAM.

Cassette - $19.95 Diskette - $24.95

BENEATH APPLE MANOR™ by Don Worth.
Descend beneath Apple Manor into an underground maze of corridors, rooms, and secret passages in quest of rich and powerful treasures. The dungeon complex consists of many floors, each lower level more dangerous than the last. If you can reach the lowest level, you may even find the ultimate object of your quest, the fabled golden apple of Apple Manor. Strategy is extremely important as you deal with a variety of monsters, each with its own characteristics. Written in integer BASIC with machine language subroutines. Requires integer BASIC and at least 16K of RAM on cassette or 32K of RAM on diskette.

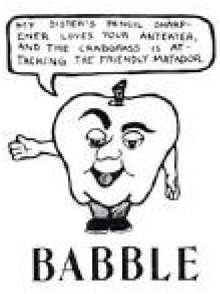
Cassette - $14.95 Diskette - $19.95

BATTLESHIP COMMANDER™ by Erik Kilk and Matthew Jew.

A game of strategy. You and the computer each start out by positioning five ships of different sizes on a ten by ten grid. Then the shooting starts. Place your volleys skillfully — a combination of logic and luck are required to beat the computer. Cartoons show the ships sinking and announce the winner. Sound effects and flashing lights also add to the enjoyment of the game. Both Applesoft and integer BASIC versions are included. Requires at least 32K of RAM.

Cassette - $14.95 Diskette - $19.95

BABBLE™ by Don Worth.
Have fun with this unique software. You write a story, entering it as a BABBLE program. As you write the story you specify certain words to be selected by the computer or entered from the keyboard at execution time. Run the program and watch BABBLE convert your story into an often hilarious collection of incongruities. The ways in which BABBLE can entertain you are limited only to your imagination. You can compose an impressive political speech or write poetry. You can plan a dinner menu. You can even form images on the screen or compose musical tunes with the help of BABBLE. The cassette version requires at least 16K of RAM and the diskette version requires at least 32K of RAM. BABBLE is written in machine language and runs on any Apple II computer.


MY SISTER'S PENCIL SHARP-ENER LOVES YOUR ANTEATER, AND THE CRABGRASS IS AT-TACKING THE FRIENDLY MATADOR.
BABBLE

Cassette - $19.95 Diskette - $24.95

L
I
N
K
E
R

LINKER by Don Worth.
Turn your Apple II or Apple II Plus into a powerful and productive software development machine with this superb linking loader/editor package. LINKER does the following and much more:

- Dynamically loads and relocates suitably prepared machine language programs anywhere in RAM.
- Combines a main program with subroutines. You can assemble a subroutine once and then use it with as many main programs as you wish.
- Produces a map of all loaded routines, giving their location and the total length of the resulting module.
- Contains a library of subroutines including binary multiplication and division, print text strings, delay, tone generator, and random number generator.

Linker works with virtually any assembler for the Apple II. Requires 32K of RAM and one disk drive.

Diskette - $49.95
Manual Only - $19.95

QS QUALITY SOFTWARE
6660 Reseda Blvd., Suite 105, Reseda, CA 91335
(213) 344-6599

Now exclusive distributors for products from The Software Factory, Newhall, California
*Apple II and Apple II Plus are trademarks of Apple Computer, Inc.

WHERE TO GET IT: Call us at (213) 344-6599 for the name of the Quality Software dealer nearest you. If necessary you may order directly from us. Mastercard and Visa cardholders may place orders by telephone. Or mail your check or bankcard number to Quality Software, 6660 Reseda Blvd., Suite 105, Reseda, CA 91335. California residents add 6% sales tax. SHIPPING CHARGES: Within North America orders must include $1.50 for first class shipping and handling. Outside North America the charge for airmail shipping and handling is $5.00. Pay in U.S. currency.

12288-13311 ($3000-$33FF). The bogus Applesoft program within it says "CALL 12304 : END", and references to locations within the program are $2800 bytes higher than shown in the listing of the version for ROM Applesoft.

```
LOC  CODE          LINE

0800               ; R.O.M. APPLESOFT REQUIRED FOR THIS.
0800               ;
0800                    *=$800
0800               BEGIN=*                    ;LOADER STARTS
0800               ;
0800               ;-----> BOGUS APPLESOFT PROGRAM:
0800               ;
0800               ; 65535 CALL 2064:END
0800               ;
0800 00                 .BYT 0          ;TRADITION
0801 0D08               .WOR $080D      ;ADDR CHAIN
0803 FFFF               .WOR $FFFF      ;LINE # 65535
0805 8C                 .BYT $8C        ;TOKEN FOR 'CALL'
0806 3230               .BYT '2064:'    ;PLACE CALLED
0808 36343A
080B 80                 .BYT $80        ;TOKEN FOR 'END'
080C 00                 .BYT 0,0,0,0    ;END OF PGM
080D 00
080E 00
080F 00
0810               ;

0810                .FILE 'PETLOAD1'
0810               ;
0810               ; I ENABLE AN APPLE II COMPUTER
0810               ; TO LOAD PET PROGRAM CASSETTES.
0810               ;
0810               ; COPYRIGHT (C) 1980
0810               ; BY: KEITH FALKNER    (416) 466-3734
0810               ;      76 ROOSEVELT ROAD
0810               ;      TORONTO CANADA  M4J 4T7
0810               ;
0810               ;-----> PAGE-ZERO WORKAREAS
0810               ;
0810               ERRORS=$A5                 ;# BAD BYTES & PARITY
0810               QUOTE=$A6                  ;AM I INSIDE QUOTES?
0810               LINSIZ=$A7                 ;BYTES IN BASIC LINE
0810               GRAPHO=$A8                 ;TRANSLATE GRAPHICS?
0810               MEMLO=$06                  ;WHERE I AM
0810               MEMHI=$07                  ; LOADING
0810               MAXLO=$08                  ;END OF
0810               MAXHI=$09                  ; PGM.
0810               ;
0810               BYTE=$18                   ;INPUT BYTE
0810               ECHO=$19                   ;ITS INVERSE
0810               PARITY=$1A                 ;PET PARITY IS ODD
0810               MAX=$1B                    ;TIME LONG BITS
0810               MIN=$1C                    ;AND SHORT BITS
0810               RETRY=$1D                  ;$80 IF RETRYING
0810               SYNCBX=$1E                 ;LOCATES A BLOCK
0810               STACK=$1F                  ;FOR FINAL RETURN ONLY
0810               ;
0810               ;-----> LABEL INPUT AREA
0810               ;
0810               LABEL=$230                 ;THIS SHOWS DATA TYPE:
0810               ; 1=LABEL OF DATA FILE
0810               ; 2=BLOCK OF DATA FILE
0810               ; 4=LABEL OF BASIC PGM
0810               ;
0810               LMEMLO=LABEL+1             ;START OF
0810               LMEMHI=LABEL+2             ; PROGRAM
0810               LMAXLO=LABEL+3             ;END OF
0810               LMAXHI=LABEL+4             ; PROGRAM
0810               LNAME=LABEL+5              ;NAME OF PGM
0810               ;
0810               ;-----> INITIALIZE
0810               ;
0810               BASIC=BEGIN+$400           ;PET PGM STARTS
0810               START=*
0810 BA                 TSX             ;FOR FINAL RETURN
0811 861F               STX STACK
0813 202FFB             JSR $FB2F       ;INIT TEXT
0816 2058FC             JSR $FC58       ;CLEAR SCREEN
0819 A911               LDA #17         ;HAND MADE
081B 8525               STA $25         ; VTAB 17
081D A98D               LDA #$8D        ;<CR> MOVES
081F 20F6FD             JSR $FDF6       ;CURSOR THERE
0822               ;
0822 A94C               LDA #$4C        ;JMP OPCODE
0824 8DF503             STA $3F5        ;RESTART &-SPOT
0827 A910               LDA #<START
0829 8DF603             STA $3F6
082C A908               LDA #>START
082E 8DF703             STA $3F7
0831               ;
0831               ; CRUCIAL TIMING FIGURES!
0831               ;
0831 A925               LDA #$25        ;> THIS MEANS 'LONG'
0833 851B               STA MAX
0835 A919               LDA #$19        ;< THIS MEANS 'SHORT'
0837 851C               STA MIN
0839               ;
0839               ;-----> POKE GREETING & RULES TO CRT
0839               ;
0839 A000               LDY #0
083B B9CC0A HELLO  LDA GREET,Y
083E 0980               ORA #$80
0840 990005             STA $500,Y
0843 B9040B             LDA RULES,Y
0846 0980               ORA #$80
0848 990006             STA $600,Y
084B B92C0B             LDA AGAIN,Y
084E 0980               ORA #$80
0850 992806             STA $628,Y
0853 C8                 INY
0854 C028               CPY #40         ;LINE FULL?
0856 D0E3               BNE HELLO
0858               ;
0858               ;-----> WAIT FOR A KEYSTROKE
0858               ;
0858 2C10C0             BIT $C010       ;NO KEY YET!
085B A960               LDA #$60        ;FLASHER
085D 8D0007             STA $700
0860 AD00C0 WAIT   LDA $C000        ;ANY KEY HIT?
0863 10FB               BPL WAIT        ;NO=WAIT
0865 48                 PHA
0866 A9A0               LDA #$A0        ;BLANK
0868 8D0007             STA $700
086B A000               LDY #0
086D 990006 WCLR   STA $600,Y       ;CLEAR RULES
0870 C8                 INY
0871 C028               CPY #40
0873 90F8               BCC WCLR
0875 2C10C0             BIT $C010       ;RESET STROBE
0878 68                 PLA
0879 C983               CMP #$83        ;CONTROL-C?
087B D001               BNE OPEN1       ;NO=GO AHEAD
087D 60                 RTS             ;ABORTED!
087E               ;
087E               ;-----> READ LABEL OF PET TAPE
087E               ;
087E A900   OPEN1  LDA #0           ;1ST TRY
0880 851D   OPEN   STA RETRY
0882 20520A             JSR BLOCK       ;FRONT OF IT
0885 A000               LDY #0          ;RESET INDEX
0887 20300A OPBYTE JSR RDBYTE       ;GET A BYTE
088A 993002             STA LABEL,Y     ;STASH BYTE
088D C005               CPY #5          ;BEFORE NAME?
088F 9005               BCC OPSTEP      ;YES=GET MORE
0891 0980               ORA #$80        ;SIGN BIT ON
0893 99FB06             STA $6FB,Y      ;PRINT NAME
0896 C8     OPSTEP INY              ;NEXT BYTE
0897 C016               CPY #22         ;21 ALREADY?
0899 90EC               BCC OPBYTE      ;N=GET MORE
089B               ;
089B               ; LABEL IS 192 BYTES, I NEED 21.
089B               ;
089B A5A5               LDA ERRORS      ;HOW DID I DO?
089D F00D               BEQ OPTYPE      ;0=PERFECT!
089F A51D               LDA RETRY       ;1ST TRY?
08A1 3004               BMI OPFAIL      ;NO
08A3 A980               LDA #$80        ;MEANS 2ND TRY
08A5 30D9               BMI OPEN
08A7 A902   OPFAIL LDA #2           ;ERROR-TYPE 2
08A9 4CCC0A OPEXIT JMP CRASH        ;ABORT!
08AC 851D   OPTYPE STA RETRY        ;0=1ST TRY
08AE AD3002             LDA LABEL       ;TYPE OF FILE
08B1 C901               CMP #1          ;BASIC PGM?
08B3 F004               BEQ SIZTST      ;YES=CONTINUE
08B5 A903               LDA #3          ;NOT BASIC
08B7 D0F0               BNE OPEXIT      ;ABORT
08B9               ;
08B9               ;-----> WILL PROGRAM FIT IN APPLE?
08B9               ;
08B9 AD3202 SIZTST LDA LMEMHI       ;PGM START
08BC C904               CMP #4          ;AT 04XX?
08BE D007               BNE SIZODD      ;NO
08C0 AD3102             LDA LMEMLO      ;DOES XX =
08C3 C902               CMP #2          ;00 OR 01?
08C5 9004               BCC SIZMAX
08C7 A906   SIZODD LDA #6           ;FUNNY ADDR
08C9 D0DE               BNE OPEXIT      ;ABORT!
08CB AD3302 SIZMAX LDA LMAXLO       ;END OF PGM
08CE 8508               STA MAXLO       ;SAVE IN PG 0
08D0 AD3402             LDA LMAXHI
08D3 18                 CLC
08D4 6908               ADC #>BEGIN
08D6 8509               STA MAXHI
08D8 C974               CMP #$74        ;FP HIMEM HI
08DA 900C               BCC LOAD        ;YES IT FITS
08DC D006               BNE SIZERR      ;NO IT DOESN'T
08DE A508               LDA MAXLO       ;IT MIGHT FIT
08E0 C973               CMP #$73        ;FP HIMEM LO
08E2 9004               BCC LOAD        ;IT JUST FITS
08E4 A904   SIZERR LDA #4           ;ERROR CODE 4
08E6 D0C1   SIZEXT BNE OPEXIT       ;ABORT!
08E8               ;
08E8               ;-----> LOAD THE PROGRAM IMAGE
08E8               ;
08E8 A90C   LOAD   LDA #>BASIC      ;INIT ADDR TO
08EA 8507               STA MEMHI       ;LOAD PGM INTO
08EC A900               LDA #$00        ;BASIC AREA
08EE 8506               STA MEMLO
08F0 A000               LDY #0
08F2 84A8               STY GRAPHO      ;SET TO TRANSLATE
08F4 9106               STA (MEMLO),Y   ;0 NEEDED THERE
08F6 85A7               STA LINSIZ
08F8               ;
08F8               ;-----> BEGIN IMAGE BLOCK
08F8               ;
08F8 20520A LDBLOK JSR BLOCK        ;BEGIN BLOCK
08FB AD00C0             LDA $C000       ;HEY, ANY KEYS?
08FE C998               CMP #$98        ;CTRL-X TO BYPASS?
0900 D003               BNE LDGO        ;NO:. LOAD IT.
```

```
0902 4C1000        JMP START       ;NEXT PGM
0905 20000A LDGO   JSR RDBYTE       ;READ THE 1ST BYTE
0908 C901          CMP #1          ;2ND LABEL?
090A 900E          BCC LDNOTZ      ;BYPASS POSSIBLE ZERO
090C D008          BNE LD1ST       ;N: BASIC IMAGE
090E 241D          BIT RETRY       ;LOST?
0910 10E6          BPL LDBLOK      ;N: OK
0912 A905          LDA #5          ;ERROR-CODE 5
0914 D0D0          BNE SIZEXT      ;ABORT!
0916 E606   LD1ST  INC MEMLO        ;BASIC+1
0918 D00D          BNE LDMOVE      ;(ALWAYS)
091A        ;
091A        ;-----> LLONG LLOADING LLOOP!
091A        ;
091A E606   LDNOTZ INC MEMLO        ;STEP TO $0C01/$3401
091C 20000A LDBYTE JSR RDBYTE
091F 241D          BIT RETRY       ;2ND TRY?
0921 1004          BPL LDMOVE      ;NO
0923 C9FC          CMP #$FC        ;BAD NOW?
0925 F004          BEQ LDSHOW      ;YES
0927 A000   LDMOVE LDY #0           ;NO INDEXING
0929 9106          STA (MEMLO),Y   ;STASH BYTE
092B 48     LDSHOW PHA
092C A506          LDA MEMLO       ;WHERE IT WENT
092E 291F          AND #$1F        ;LIMIT SPREAD
0930 AA            TAX             ;PREP INDEX
0931 68            PLA
0932 48            PHA
0933 0980          ORA #$80
0935 9D2C04        STA $42C,X      ;SHOW THE BYTE
0938 E6A7          INC LINSIZ
093A A905          LDA #5          ;IN 1ST 4 BYTES,
093C C5A7          CMP LINSIZ      ;IGNORE 00 & CF.
093E 900D          BCC LDPULL
0940 68            PLA
0941 AA            TAX             ;END-OF-LINE?
0942 D002          BNE LDLOWO      ;NO
0944 85A7          STA LINSIZ      ;RESET
0946 C9CF   LDLOWO CMP #$CF         ;LOWER CASE O?
0948 D004          BNE LDDONE
094A 85A8          STA GRAPHQ      ;NO TRANSLATION
094C 48            PHA
094D 68     LDPULL PLA
094E        ;
094E        ;-----> HAVE I DONE IT ALL?
094E        ;
094E A507   LDDONE LDA MEMHI        ;WHERE IT IS
0950 C509          CMP MAXHI       ;END OF PGM
0952 D006          BNE LDSTEP      ;MORE TO DO
0954 A506          LDA MEMLO
0956 C508          CMP MAXLO
0958 F008          BEQ LDTEST
095A E606   LDSTEP INC MEMLO        ;STEP 1 BYTE
095C D0BE          BNE LDBYTE      ;SAME PAGE
095E E607          INC MEMHI       ;NEW PAGE
0960 D0BA          BNE LDBYTE      ;(ALWAYS)
0962        ;
0962        ;-----> HOW WELL DID IT LOAD?
0962        ;
0962 A5A5   LDTEST LDA ERRORS       ;# BAD BYTES
0964 F00B          BEQ LDING       ;NONE!!!!!!!
0966 A51D          LDA RETRY       ;2ND PASS?
0968 3007          BMI LDING       ;YES
096A A980          LDA #$80        ;NOW IT IS!
096C 851D          STA RETRY
096E 4CE808        JMP LOAD
0971 20E2FB LDING  JSR $FBE2        ;RING BELL
0974        ; BELL MEANS 'DONE WITH TAPE'.
0974        ;
0974        ;-----> PREPARE PAGE ZERO POINTERS
0974        ;       FOR APPLESOFT INTERPRETER.
0974        ;
0974 A901          LDA #$01
0976 8567          STA $67         ;START OF BASIC
0978 A90C          LDA #>BASIC
097A 8568          STA $68
097C A608          LDX MAXLO
097E 86AF          STX $AF         ;END OF
0980 A409          LDY MAXHI       ;PROGRAM.
0982 84B0          STY $B0
0984 E8            INX
0985 D001          BNE POINT1
0987 C8            INY
0988 E8     POINT1 INX              ;2 ABOVE THAT
0989 D001          BNE POINT2      ;IS START OF
098B C8            INY             ;BOTH ...
098C 8669   POINT2 STX $69
098E 846A          STY $6A         ;VARIABLES
0990 866B          STX $6B
0992 846C          STY $6C         ;& ARRAYS.
0994 866D          STX $6D
0996 846E          STY $6E         ;END OF VARS.
0998        ;
0998        ;-----> ADJUST THE ADDRESS-CHAIN,
0998        ;       BECAUSE THIS PROGRAM LIVED
0998        ;       AT $0401 IN THE PET, AND
0998        ;       WAS LOADED JUST ABOVE ME.
0998        ;
0998 A567          LDA $67
099A 8506          STA MEMLO       ;START OF
099C A568          LDA $68         ;ADDR-CHAIN
099E 8507          STA MEMHI
09A0 A000   ADJUST LDY #0           ;NO INDEXING!
09A2 B106          LDA (MEMLO),Y   ;LO ADDR LINK
09A4 AA            TAX             ;HOLD IT!
09A5 C8            INY             ;INDEX BY 1
09A6 B106          LDA (MEMLO),Y   ;HI ADDR LINK
09A8 F00B          BEQ CONVRT      ;0=ALL DONE
09AA 18            CLC
09AB 6908          ADC #>BEGIN
09AD 9106          STA (MEMLO),Y   ;PUT IT BACK
09AF 8606          STX MEMLO       ;UPDATE BOTH
09B1 8507          STA MEMHI       ;LINK BYTES.
09B3 D0EB          BNE ADJUST      ;(ALWAYS)
09B5        ;
09B5        ;-----> NOW TO CONVERT TOKENS
09B5        ;
09B5 A567   CONVRT LDA $67
09B7 8506          STA MEMLO       ;START OF
09B9 A568          LDA $68         ;PROGRAM.
09BB 8507          STA MEMHI
09BD        ;
09BD A000   COSCAN LDY #0           ;NO INDEXING!
09BF B106          LDA (MEMLO),Y
09C1 AA            TAX
09C2 C8            INY
09C3 B106          LDA (MEMLO),Y
09C5 F078          BEQ APPLE       ;IF DONE CONV.
09C7 88            DEY             ;Y=0 FROM HERE DOWN!
09CD        ;
09C8 84A6          STY QUOTE       ;NOT IN QUOTES
09CA 8608          STX MAXLO       ;-> FINISH OF
09CC 8509          STA MAXHI       ;CURRENT LINE.
09CE        ;
09CE 18            CLC
09CF A506          LDA MEMLO       ;STEP PAST
09D1 6904          ADC #4          ;ADDR-CHAIN
09D3 8506          STA MEMLO       ;& LINE NO.
09D5 A507          LDA MEMHI
09D7 6900          ADC #0
09D9 8507          STA MEMHI
09DB        ;
09DB B106   COBYTE LDA (MEMLO),Y    ;PICK A BYTE
09DD C922          CMP #$22        ;IS IT QUOTE?
09DF F043          BEQ COQUOT      ;YUP.
09E1 24A6          BIT QUOTE       ;IN QUOTES?
09E3 3016          BMI CODATA      ;YUP
09E5        ;
09E5        ; IS THIS BYTE A PET TOKEN?
09E5        ;
09E5 C9FC          CMP #$FC        ;BADLY READ?
09E7 F041          BEQ COSTEP      ;YES. KEEP IT.
09E9 C9CB          CMP #$CB        ;>LARGEST PET TOKEN?
09EB B008          BCS COUNDF      ;YES: SUBSTITUTE.
09ED AA            TAX             ;PREP INDEX
09EE 103A          BPL COSTEP      ;IF NOT TOKEN
09F0 BD2C0B        LDA TOKENS-$80,X ;APPLE TOKEN
09F3 D002          BNE COTOKE      ;ALWAYS
09F5 A9FB   COUNDF LDA #$FB         ;UNDEF'D FUNCTION
09F7 9106   COTOKE STA (MEMLO),Y    ;FIX PET TOKEN
09F9 D02F          BNE COSTEP      ;ALWAYS
09FB        ;
09FB        ; DEAL WITH DATA INSIDE QUOTES:
09FB        ; SOME ARE TRANSLATED. THE REST
09FB        ; HAVE THEIR SIGN-BITS REMOVED.
09FB        ;
09FB A210   CODATA LDX #GRAFIC-DATRAN
09FD CA     CODTRY DEX
09FE DD530B        CMP DATRAN-1,X ;TEST BYTE
0A01 F019          BEQ CODSUB      ;YES: TRANSLATE IT.
0A03 CA            DEX             ;STEP BACK
0A04 D0F7          BNE CODTRY      ;IF MORE TO TEST
0A06        ;  THE BYTE WAS NOT IN 'DATRAN'.
0A06        ;  SEE IF I TRANSLATE GRAPHICS.
0A06 A6A8          LDX GRAPHQ      ;ANY LOWER-CASE 'O'?
0A08 D00E          BNE COPLUS      ;YES. NO TRANSLATE.
0A0A C9E0          CMP #$E0        ;GRAPHIC KEY?
0A0C B00A          BCS COPLUS      ;NO
0A0E C9A0          CMP #$A0        ;FOR SURE?
0A10 9006          BCC COPLUS      ;NO
0A12 E9A0          SBC #$A0        ;CALC INDEX
0A14 AA            TAX
0A15 BD6C0B        LDA GRAFIC,X
0A19 297F   COPLUS AND #$7F         ;KILL SIGN
0A1A 1003          BPL CODSTA      ;ALWAYS
0A1C BD540B CODSUB LDA DATRAN,X     ;GET SUBSTITUTE
0A1F 9106   CODSTA STA (MEMLO),Y    ;REPLACE DATA.
0A21 4C2A0A CONEXT JMP COSTEP
0A24        ;
0A24        ;-----> FOUND QUOTE, SO FLIP SWITCH.
0A24        ;
0A24 A5A6   COQUOT LDA QUOTE
0A26 4980          EOR #$80        ;INVERT SIGH
0A28 85A6          STA QUOTE
0A2A        ;
0A2A        ; STEP TO NEXT BYTE IN LINE.
0A2A        ;
0A2A E606   COSTEP INC MEMLO
0A2C D002          BNE COTEST
0A2E E607          INC MEMHI
0A30 A506   COTEST LDA MEMLO
0A32 C508          CMP MAXLO       ;DONE A LINE?
0A34 D0A5          BNE COBYTE      ;N: NEXT BYTE
0A36 A507          LDA MEMHI
0A38 C509          CMP MAXHI
0A3A D09F          BNE COBYTE
0A3C 4CBD09        JMP COSCAN      ;Y: NEXT LINE
0A3F        ;
0A3F        ;-----> FINISHED! PRINT 'OK' (CR)
0A3F        ;
0A3F A9CF   APPLE  LDA #$CF         ;'O'
0A41 20F6FD        JSR $FDF6
0A44 A9CB          LDA #$CB        ;'K'
0A46 20F6FD        JSR $FDF6
0A49 A98D          LDA #$8D        ;(CR)
0A4B 20F6FD        JSR $FDF6
0A4E        ;
0A4E        ;-----> THE ONLY WAY OUT
```

```
0A4E               ;
0A4E A61F    EXIT   LDX STACK
0A50 9A             TXS
0A51 60             RTS
0A52               ;

0A52                .FILE 'PETLOAD2'
0A52               ;
0A52               ;-----> POSITION TO START OF BLOCK
0A52               ;
0A52 A900    BLOCK  LDA #0          ;ZERO COUNT
0A54 85A5           STA ERRORS      ;OF ERRORS
0A56 20800A  BLPASS JSR RDBYTE      ;GET A BYTE
0A59 451D           EOR RETRY       ;(OR 04)
0A5B C984           CMP #$84        ;HEX 84?
0A5D F008           BEQ BLSYNC      ;FOUND IT!
0A5F               ;
0A5F               ; EACH BLOCK BEGINS THUS:
0A5F               ; 89,88,87,86,85,84,83,82,81, DATA
0A5F               ; (DUPLICATES ARE 09,08,...)
0A5F               ;
0A5F               ; I LOOK FOR $84, THEN THE REST.
0A5F               ;
0A5F 24A5           BIT ERRORS      ;MANY ERRORS?
0A61 10F3           BPL BLPASS      ;<128 = CONTINUE
0A63 A901    BLFAIL LDA #1          ;ERROR CODE 1
0A65 D065           BNE CRASH       ;ABORT!
0A67               ;
0A67               ; INSIST ON 83, 82, 81:
0A67               ;
0A67 851E    BLSYNC STA SYNCBX      ;SAVE THE 84
0A69 A900           LDA #0          ;FORGIVE ANY
0A6B 85A5           STA ERRORS      ;ERRORS SO FAR
0A6D               ;
0A6D C61E    BLWALK DEC SYNCBX      ;COUNTDOWN
0A6F A980           LDA #$80
0A71 C51E           CMP SYNCBX      ;PAST THE $81?
0A73 F046           BEQ RBEXIT      ;YES; EXIT.
0A75 20800A         JSR RDBYTE      ;GET NEXT
0A78 451D           EOR RETRY       ;ALLOW FOR RETRY
0A7A C51E           CMP SYNCBX      ;RIGHT BYTE?
0A7C F0EF           BEQ BLWALK      ;Y; CONTINUE
0A7E D0D6           BNE BLPASS      ;N; STROLL ON
0A80               ;
0A80               ;-----> READ A BYTE FROM TAPE
0A80               ;
0A80 20BC0A  RDBYTE JSR RDBIT       ;GET A BIT
0A83 E41B           CPX MAX         ;BIG FAT ONE?
0A85 90F9           BCC RDBYTE      ;WAIT FOR IT
0A87 20BC0A         JSR RDBIT       ;IGNORE NEXT
0A8A A980           LDA #$80        ;SET TO GET 8
0A8C 8519           STA ECHO
0A8E               ;
0A8E               ; WHEN THAT $80 FALLS OUT THE
0A8E               ; RIGHT SIDE OF 'ECHO', EIGHT
0A8E               ; BITS HAVE FLOWED INTO BYTE.
0A8E               ;
0A8E 851A           STA PARITY      ;START EVEN
0A90 20BC0A  RD8BIT JSR RDBIT       ;SET OR CLR CARRY
0A93 9002           BCC RDROR       ;IS IT A ZERO-BIT?
0A95 E61A           INC PARITY      ;COUNT ONE-BITS.
0A97 6618    RDROR  ROR BYTE        ;ASSEMBLE DATA
0A99 20BC0A         JSR RDBIT       ;GET PARTNER
0A9C 6619           ROR ECHO        ;ABSORB A BIT
0A9E 90F0           BCC RD8BIT      ;GO FOR 8 BITS
0AA0 20BC0A         JSR RDBIT       ;READ PARITY
0AA3 9002           BCC RDTEST      ;IF IT'S ZERO
0AA5 E61A           INC PARITY      ;ELSE COUNT
0AA7 A519    RDTEST LDA ECHO
0AA9 49FF           EOR #$FF        ;INVERT ECHO
0AAB C518           CMP BYTE        ;MUST = BYTE
0AAD D004           BNE RDBAD       ;NOPE.
0AAF 661A           ROR PARITY      ;ODD PARITY?
0AB1 B007           BCS RDEXIT      ;YUP.
0AB3 E6A5    RDBAD  INC ERRORS
0AB5 A9FC           LDA #$FC        ;TOKEN FOR ERROR
0AB7 8D30C0         STA $C030       ;TOGGLE SPEAKER
0ABA AA      RDEXIT TAX             ;SET FLAGS Z/N
0ABB 60      RBEXIT RTS
0ABC               ;
0ABC               ;-----> READ A BIT FROM TAPE.
0ABC               ;
0ABC               ; THERE ARE, BELIEVE IT OR NOT,
0ABC               ; 4 TRANSITIONS IN A BIT. THIS
0ABC               ; CODE IGNORES POSITIVE ONES,
0ABC               ; BECAUSE THE NEGATIVE ONES ARE
0ABC               ; MORE DRAMATIC. NOW, A LONG AND
0ABC               ; A MEDIUM STARTS A BYTE, AFTER
0ABC               ; WHICH A SHORT AND A MEDIUM IS
0ABC               ; A ZERO-BIT, WHILE A MEDIUM AND
0ABC               ; A SHORT IS A ONE-BIT. ALL THE
0ABC               ; NEGATIVE TRANSITIONS ARE READ
0ABC               ; AND TESTED, AND ONLY VALID BIT
0ABC               ; COMBINATIONS ARE PERMITTED.
0ABC               ;
0ABC A200    RDBIT  LDX #0          ;START TIMER
0ABE 2C60C0  RDBITW BIT $C060       ;LOOK @ INPUT
0AC1 10FB           BPL RDBITW      ;IGNORE +VE.
0AC3 E8      RDBITC INX             ;TIME -VE WAVE
0AC4 2C60C0         BIT $C060       ;LOOK @ INPUT
0AC7 30FA           BMI RDBITC      ;WAIT FOR +VE
0AC9 E41C           CPX MIN         ;SET CARRY BIT
0ACB 60             RTS
0ACC               ;
0ACC               ;-----> DISASTER! ACCUM CONTAINS:
0ACC               ;
0ACC               ;  1 -- CAN'T FIND START OF LABEL
0ACC               ;  2 -- BAD BITS/PARITY IN LABEL
0ACC               ;  3 -- THIS ISN'T A BASIC PGM
0ACC               ;  4 -- PGM IS TOO BIG FOR APPLE
0ACC               ;  5 -- CAN'T FIND START OF PGM
0ACC               ;  6 -- IT LOADS @ A FUNNY ADDR
0ACC               ;
0ACC 09B0    CRASH  ORA #$B0        ;NOW ASCII
0ACE 20F6FD         JSR $FDF6       ;PRINT IT.
0AD1 A9A0           LDA #$A0        ;MAKE A SPACE
0AD3 20F6FD         JSR $FDF6       ;PRINT IT TOO
0AD6 202DFF         JSR $FF2D       ;'ERR' & BELL
0AD9 4C4E0A         JMP EXIT        ;EXIT
0ADC               ;
0ADC 5020    GREET  .BYT 'P E T  L O A D E R '
0ADE 4520
0AE0 5420
0AE2 2020
0AE4 4C20
0AE6 4F20
0AE8 4120
0AEA 4420
0AEC 4520
0AEE 5220
0AF0 2020           .BYT '    BY FALKNER  V2.0'
0AF2 2020
0AF4 4259
0AF6 2046
0AF8 414C
0AFA 4B4E
0AFC 4552
0AFE 2020
0B00 5632
0B02 2E30
0B04 504C    RULES  .BYT 'PLAY PET PROGRAM TAP'
0B06 4159
0B08 2050
0B0A 4554
0B0C 2050
0B0E 524F
0B10 4752
0B12 414D
0B14 2054
0B16 4150
0B18 4520           .BYT 'E AND PRESS ANY KEY.'
0B1A 414E
0B1C 4420
0B1E 5052
0B20 4553
0B22 5320
0B24 414E
0B26 5920
0B28 4B45
0B2A 592E
0B2C 2852    AGAIN  .BYT '(RESTART FROM APPLES'
0B2E 4553
0B30 5441
0B32 5254
0B34 2046
0B36 524F
0B38 4D20
0B3A 4150
0B3C 504C
0B3E 4553
0B40 4F46           .BYT 'OFT BY TYPING ''&''.) '
0B42 5420
0B44 4259
0B46 2054
0B48 5950
0B4A 494E
0B4C 4720
0B4E 2726
0B50 272E
0B52 2920
0B54               ;
0B54               DATRAN=*              ;DATA TO TRANSLATE:
0B54 118A               .DBY $118A      ;DOWN       - LINEFEED
0B56 129E               .DBY $129E      ;RVS        - INVERSE
0B58 1397               .DBY $1397      ;HOME       - HOME
0B5A 1485               .DBY $1485      ;DEL        - DEL
0B5C 1DA0               .DBY $1DA0      ;RIGHT      - SPACE
0B5E 918F               .DBY $918F      ;UP         - VLIN
0B60 929D               .DBY $929D      ;OFF        - NORMAL
0B62 938D               .DBY $938D      ;CLR        - CLEAR
0B64 9488               .DBY $9488      ;INST       - IN#
0B66 9D88               .DBY $9D88      ;LEFT       - BACKSPACE
0B68 FCFC               .DBY $FCFC      ;ERROR BEEP - ERROR BEEP
0B6A A227               .DBY $A227      ;SHIFTQUOTE - APOSTROPHE
0B6C               ;
0B6C               GRAFIC=*              ;APPROX GRAPHIC KEYS
0B6C 2058               .BYT ' X'       ;$A0-A1
0B6E 3D2D               .BYT '=-'       ;$A2-A3  I HAVE TO MAKE
0B70 5F28               .BYT '_('       ;$A4-A5  A GUESS:
0B72 2A29               .BYT '*)'       ;$A6-A7
0B74 3D2F               .BYT '=/'       ;$A8-A9  DOES THE PROGRAM
0B76 5D21               .BYT ']!'       ;$AA-AB  DISPLAY LOWER-
0B78 2E2E               .BYT '..'       ;$AC-AD  CASE LETTERS, OR
0B7A 2E3D               .BYT '.='       ;$AE-AF  DOES IT USE THE
0B7C 2E2D               .BYT '.-'       ;$B0-B1  PET GRAPHICS
0B7E 2D21               .BYT '-!'       ;$B2-B3  CHARACTERS?
0B80 5B5B               .BYT '[['       ;$B4-B5
0B82 5D3D               .BYT ']='       ;$B6-B7  I LOOK FOR A
0B84 3D3D               .BYT '=='       ;$B8-B9  LOWER-CASE 'O'
0B86 5D2E               .BYT '].'       ;$BA-BB  WHILE LOADING.
0B88 2E2E               .BYT '..'       ;$BC-BD
0B8A 2E3F               .BYT '.?'       ;$BE-BF  IF I FIND ONE,
0B8C 2D53               .BYT '-S'       ;$C0-C1  I PRESUME THAT
0B8E 212D               .BYT '!-'       ;$C2-C3  THE PROGRAM IS
0B90 2D2D               .BYT '--'       ;$C4-C5  PRINTING LOWER
0B92 2D28               .BYT '-('       ;$C6-C7  CASE LETTERS,
0B94 292E               .BYT ').'       ;$C8-C9  AND I DO NOT
0B96 2E2E               .BYT '..'       ;$CA-CB  USE THIS TABLE.
```

```
0B98 5B5C           .BYT '[\'        ;$CC-CD
0B9A 2F5B           .BYT '/['        ;$CE-CF  OF COURSE, I
0B9C 5D2A           .BYT ']*'        ;$D0-D1  MIGHT BE WRONG
0B9E 5F40           .BYT '_@'        ;$D2-D3  EITHER WAY.
0BA0 282E           .BYT '(.'        ;$D4-D5  AND INDEED SOME
0BA2 582B           .BYT 'X+'        ;$D6-D7  PROGRAMS USE
0BA4 4329           .BYT 'C)'        ;$D8-D9  THE KEYS BOTH
0BA6 442B           .BYT 'D+'        ;$DA-DB  WAYS.
0BA8 5B21           .BYT '[!'        ;$DC-DD
0BAA 235C           .BYT '#\'        ;$DE-DF
0BAC           ;
0BAC           ;-----> THE MIGHTY TOKEN TABLE!
0BAC           ;       AT (TOKENS - $80 + PET TOKEN)
0BAC           ;       IS THE SAME APPLE TOKEN.
0BAC           TOKENS=*
0BAC 80             .BYT $80         ;END
0BAD 81             .BYT $81         ;FOR
0BAE 82             .BYT $82         ;NEXT
0BAF 83             .BYT $83         ;DATA
0BB0 8B             .BYT $8B         ;INPUT#/IN#    <-HELP!
0BB1 84             .BYT $84         ;INPUT
0BB2 86             .BYT $86         ;DIM
0BB3 87             .BYT $87         ;READ
0BB4 AA             .BYT $AA         ;LET
0BB5 AB             .BYT $AB         ;GOTO
0BB6 AC             .BYT $AC         ;RUN
0BB7 AD             .BYT $AD         ;IF
0BB8 AE             .BYT $AE         ;RESTORE
0BB9 B0             .BYT $B0         ;GOSUB
0BBA B1             .BYT $B1         ;RETURN
0BBB B2             .BYT $B2         ;REM
0BBC B3             .BYT $B3         ;STOP
0BBD B4             .BYT $B4         ;ON
0BBE B5             .BYT $B5         ;WAIT
0BBF B6             .BYT $B6         ;LOAD
0BC0 B7             .BYT $B7         ;SAVE
0BC1 B3             .BYT $B3         ;VERIFY/STOP
0BC2 B8             .BYT $B8         ;DEF
0BC3 B9             .BYT $B9         ;POKE
0BC4 BA             .BYT $BA         ;PRINT#/PR#    <-HELP!
0BC5 BA             .BYT $BA         ;PRINT
0BC6 BB             .BYT $BB         ;CONT
0BC7 BC             .BYT $BC         ;LIST
0BC8 BD             .BYT $BD         ;CLEAR
0BC9 B3             .BYT $B3         ;CMD/STOP
0BCA 8C             .BYT $8C         ;SYS/CALL      <-HELP!
0BCB B3             .BYT $B3         ;OPEN/STOP
0BCC B3             .BYT $B3         ;CLOSE/STOP
0BCD BE             .BYT $BE         ;GET
0BCE BF             .BYT $BF         ;NEW
0BCF C0             .BYT $C0         ;TAB(
0BD0 C1             .BYT $C1         ;TO
0BD1 C2             .BYT $C2         ;FN
0BD2 C3             .BYT $C3         ;SPC(
0BD3 C4             .BYT $C4         ;THEN
0BD4 C6             .BYT $C6         ;NOT
0BD5 C7             .BYT $C7         ;STEP
0BD6 C8             .BYT $C8         ;+
0BD7 C9             .BYT $C9         ;-
0BD8 CA             .BYT $CA         ;*
0BD9 CB             .BYT $CB         ;/
0BDA CC             .BYT $CC         ;↑
0BDB CD             .BYT $CD         ;AND
0BDC CE             .BYT $CE         ;OR
0BDD CF             .BYT $CF         ;>
0BDE D0             .BYT $D0         ;=
0BDF D1             .BYT $D1         ;<
0BE0 D2             .BYT $D2         ;SGN
0BE1 D3             .BYT $D3         ;INT
0BE2 D4             .BYT $D4         ;ABS
0BE3 D5             .BYT $D5         ;USR
0BE4 D6             .BYT $D6         ;FRE
0BE5 D9             .BYT $D9         ;POS
0BE6 DA             .BYT $DA         ;SQR
0BE7 DB             .BYT $DB         ;RND
0BE8 DC             .BYT $DC         ;LOG
0BE9 DD             .BYT $DD         ;EXP
0BEA DE             .BYT $DE         ;COS
0BEB DF             .BYT $DF         ;SIN
0BEC E0             .BYT $E0         ;TAN
0BED E1             .BYT $E1         ;ATN
0BEE E2             .BYT $E2         ;PEEK
0BEF E3             .BYT $E3         ;LEN
0BF0 E4             .BYT $E4         ;STR$
0BF1 E5             .BYT $E5         ;VAL
0BF2 E6             .BYT $E6         ;ASC
0BF3 E7             .BYT $E7         ;CHR$
0BF4 E8             .BYT $E8         ;LEFT$
0BF5 E9             .BYT $E9         ;RIGHT$
0BF6 EA             .BYT $EA         ;MID$
0BF7                .END

ERRORS = 0000 (0000)
```

```
SYMBOL TABLE

SYMBOL    VALUE

ADJUST  09A0  AGAIN   0B2C  APPLE   0A3F
BASIC   0C00  BEGIN   0800  BLFAIL  0A63
BLOCK   0A52  BLPASS  0A56  BLSYNC  0A67
BLWALK  0A6D  BYTE    0018  COBYTE  090B
CODATA  09FB  CODSTA  0A1F  CODSUB  0A1C
CODTRY  09FD  CONEXT  0A21  CONVRT  0985
COPLUS  0A18  COQUOT  0A24  COSCAN  098D
COSTEP  0A2A  COTEST  0A30  COTOKE  09F7
COUNDF  09F5  CRASH   0ACC  DATRAN  0B54
ECHO    0019  ERRORS  00A5  EXIT    0A4E
GRAFIC  0B6C  GRAPHO  00A8  GREET   0ADC
HELLO   083B  LABEL   0230  LD1ST   0916
LDBLOK  09F8  LDBYTE  091C  LDDONE  094E
LDGO    0905  LDING   0971  LDLOWO  0946
LDMOVE  0927  LDNOTZ  091A  LDPULL  094D
LDSHOW  092B  LDSTEP  095A  LDTEST  0962
LINSIZ  00A7  LMAXHI  0234  LMAXLO  0233
LMEMHI  0232  LMEMLO  0231  LNAME   0235
LOAD    08E8  MAX     001B  MAXHI   0009
MAXLO   0008  MEMHI   0007  MEMLO   0006
MIN     001C  OPBYTE  0887  OPEN    0800
OPEN1   087E  OPEXIT  08A9  OPFAIL  08A7
OPSTEP  0896  OPTYPE  00AC  PARITY  001A
POINT1  0988  POINT2  098C  QUOTE   00A6
RDEXIT  0AB0  RDGBIT  0A90  RDBAD   0AB3
RDBIT   0A8C  RDBITC  0AC3  RDBITW  0ABE
RDBYTE  0A80  RDEXIT  0ABA  RDROR   0A97
RDTEST  0AA7  RETRY   001D  RULES   0B04
SIZERR  08E4  SIZEXT  08E6  SIZMAX  08C8
SIZODD  08C7  SIZTST  08B9  STACK   001F
START   0810  SYNCSX  001E  TOKENS  0BAC
WAIT    0860  WCLR    086D
END OF ASSEMBLY
```

Apple Monitor Extender

APPLE II 16K, CASSETTE

This utility program works in complete harmony with the Apple monitor to extend your computer's capability and help you use the full power of machine language programming.

Screen display shows memory in HEX, ASCII or BINARY. Move data anywhere in memory without regard to direction or overlapping and read or write any sector on disk. Insertions may be in HEX or ASCII so you can easily format high speed text displays without conversions.

Study, modify or disassemble any program, complete with labels. Several programs may be combined, and the entire disassembled text file stored on disk/tape for later assembly. The slow listing feature steps through listings with ease.

©Copyright 1980 Glenn R. Sogge, All Rights Reserved.

IMAGE COMPUTER PRODUCTS™

615 Academy Drive
Northbrook, IL 60062
312/564-5060

Programming And Interfacing The Apple, With Experiments

Marvin L. De Jong
Department of Mathematics – Physics
The School of the Ozarks
Pt. Lookout, MO 65726

Introduction

When the Apple microcomputer is compared with other popular microcomputers, one of its most attractive features is the ease with which it can be interfaced to devices in the outside world. Particularly important in this connection are those eight beautiful card slots in the Apple. The "black box" philosophy of the designers of the TRS-80 leaves much to be desired in scientific, educational, or industrial applications.

In this article we will describe a circuit to be built on a peripheral card that fits in any of the eight card slots in the Apple. The circuit provides the user with one eight-bit input port and one eight-bit output port (with possibilities for expansion). The circuit is built with readily available components, and the output port is also attached to eight LEDs so the user can visualize the state of the bits. The bit values of the input port may be controlled with an eight element DIP switch, or by devices of the user's own choice, such as an A/D converter.

My main reason for designing this circuit was to provide Apple owners with the experiments in my book **Programming & Interfacing the 6502, With Experiments.** This book was originally based on the KIM-1, SYM-1, and the AIM 65, but with the I/O board described in this article, the book can be used in conjunction with Apple computers. So, if you are interested in learning assembly language programming in conjunction with my book, this I/O board may make the task a little easier. If you are not interested in the book, the I/O board described here will be of interest if you wish to interface your Apple computer to devices like A/D and D/A converters, stepper motors, transmitters, and a variety of other devices.

The Circuit

The circuit diagram is shown in Figure 1. The circuitry on the Apple microcomputer develops a $\overline{\text{DEVICE SELECT}}$ pulse for each of the eight peripheral cards. For card number 0, this pulse occurs whenever addresses $C080 through $C08F appear on the address bus. For card number 1, the corresponding addresses are $C090 through $C09F, and so on for the other card numbers. That is, the device select pulse, $\overline{\text{DS}}$, is at logic zero for each address in the range $C0N0 through $C0NF, where N = $8 + CN and CN is the card number expressed in hexadecimal. These device select pulses can activate up to 16 I/O ports on each card.

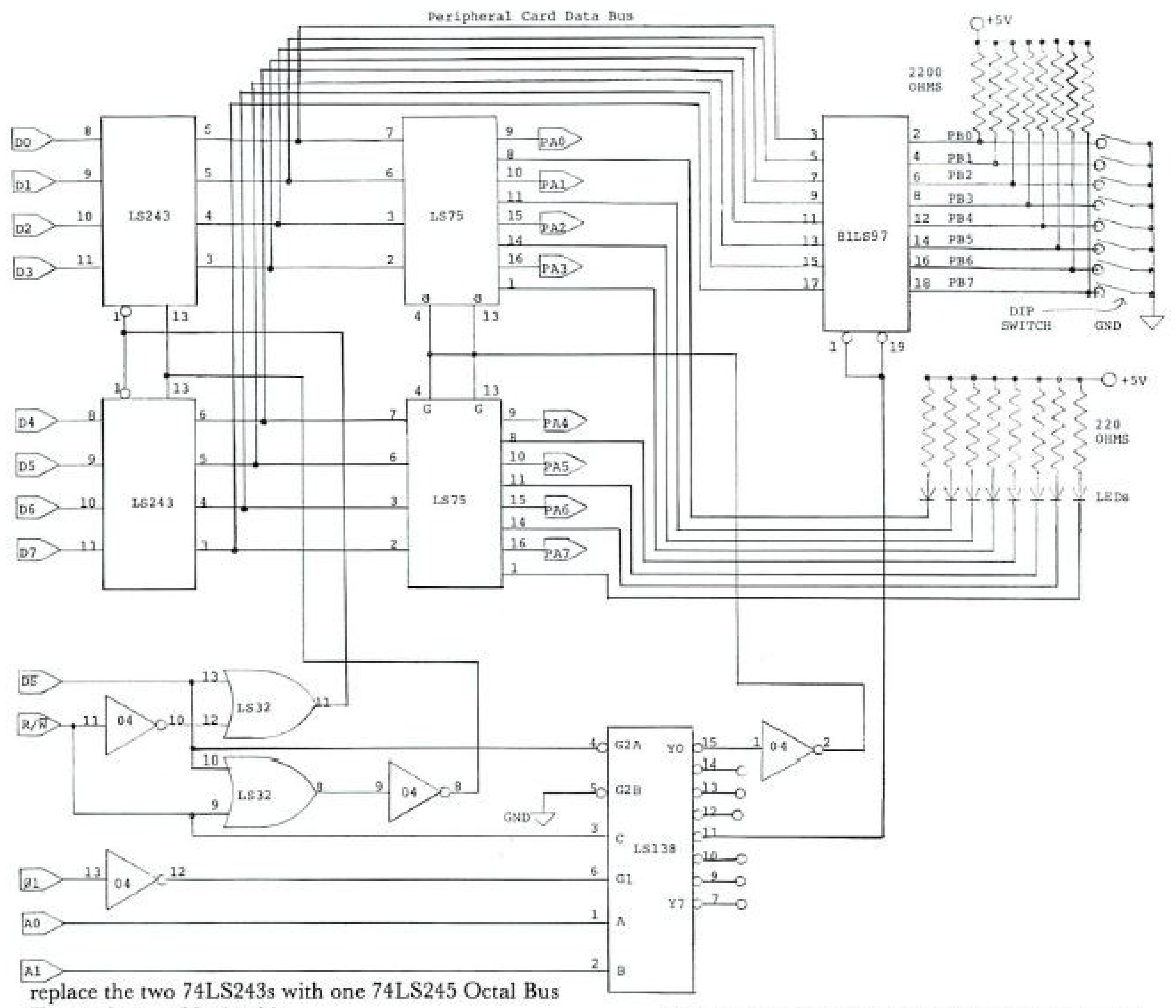
In the circuit shown in Figure 1, the device select pulse ($\overline{\text{DS}}$) generated by the Apple is combined with the R/$\overline{\text{W}}$ signal generated by the 6502 to produce a signal that activates the 74LS242 bus transceivers in the direction from the Apple data bus to a peripheral card data bus when the R/$\overline{\text{W}}$ line is at logic zero. The lower 74LS32 OR gate and the 74LS04 INVERTER generate this signal. When the R/$\overline{\text{W}}$ line is at logic one during a 6502 READ cycle, the 74LS243 bus transceivers drive the Apple data bus from the peripheral card data bus, provided the $\overline{\text{DS}}$ signal is at logic zero, its active state. Thus, all data bus buffering is handled by the two 74LS243s and it is controlled by the $\overline{\text{DEVICE SELECT}}$ pulse and the R/$\overline{\text{W}}$ line coming from the Apple computer. If you want to reduce the chip count,

Computer House Division

PROGRAMS FOR COMMODORE AND APPLE	
Legal accounting Demo	$15.00
Legal accounting Program	995.00
Machine Part Quote Demo	15.00
Machine Part Quote Program	325.00
Mailing/phone list	80.00
Political Mail/phone list	130.00
Beams, structural	115.00
Trig/Circle Tangent	110.00
Spur Gears	35.00
Bolt Circles	25.00
Filament Wound TAnks	125.00
Scrunch	25.00

PROGRAMS FOR COMMODORE ONLY	
A/P, A/R, Job Cost & Job Est.	370.00
Inventory	95.00
Financial	175.00
Real Estate Listings	265.00
Check Writer	25.00
File Editing Tools (FET)	65.00
Screen Dump/Repeat	35.00
Docu-Print	20.00
Scrunch	25.00
Sof-Bkup	40.00
Sorter (Mach. Language)	35.00
Trace-Print	25.00
Vari-Print	25.00

ASK FOR CATALOG #80-C2 Dealers Wanted
Computer House Div. 1407 Clinton Road
Jackson, Michigan 49202 (517) 782-2132

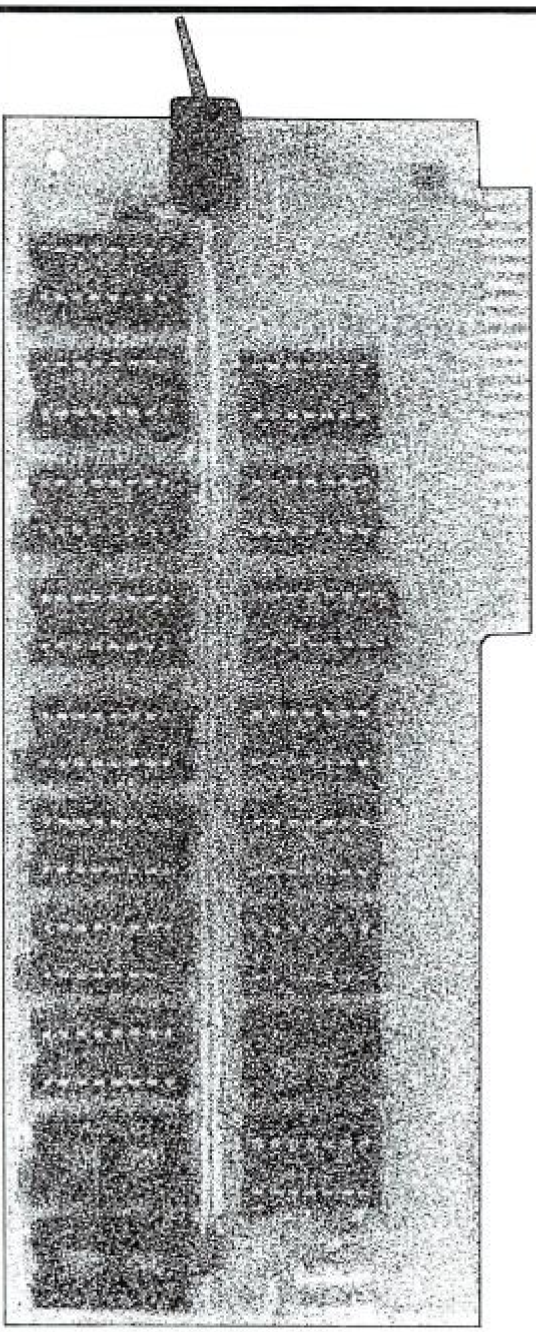


replace the two 74LS243s with one 74LS245 Octal Bus Transceiver, a 20-pin chip.

The $\overline{DS}$ pulse also activates the 74LS138 3-to-8 line decoder that is used to produce up to four $\overline{WRITE}$ pulses for output ports (Y_0 - Y_3 pins on the LS138) and four $\overline{READ}$ pulses for input ports (Y_4 - Y_7 pins on the LS138). In this application, only one $\overline{WRITE}$ pulse and one $\overline{READ}$ pulse are used. You may add more ports. Note that the R/$\overline{W}$ line is one of the lines decoded by the 74LS138. This idea originated in Gene Zumchak's Nuts and Volts column in Issue 2 of **compute II**. Thus, the lowest four output lines, Y_0 - Y_3, on the 74LS138 will be active only on WRITE cycles. The highest four output lines from the 74LS138, namely Y_4 - Y_7, will be active on READ cycles. In particular, Y_0 is active when the last nibble in the address is $0 ($C080, for example) and the 6502 is in a WRITE cycle with the R/$\overline{W}$ line at logic zero. The Y_4 line from the 74LS138 is active during a READ cycle and when the low-order nibble in the address is $4 ($C084, for example).

The two 74LS75 4-bit bistable latches are transparent to the data bus when the G inputs are at logic one. At the conclusion of the WRITE cycle the G inputs are brought to logic zero and the data on the data bus are latched into the Q outputs of the two 74LS75s. These eight pins for the output port we have called PORT A. The Q outputs activate the LEDs in the sense that the LED will glow if the bit value in Port A is one; otherwise they will not glow. If you want to reduce the chip count, replace the 74LS75s with octal latches such as the 74LS363 or 373, but you will have to eliminate the pretty LEDs or find another way to drive them. This completes the description of the output port.

The input port is an 81LS97 octal three-state buffer. It drives the peripheral card data bus when the Y_4 line on the 74LS138 is at logic zero. Thus, a LDA $C094 will result in a "read" of the input port if the peripheral card is in slot 1 of the Apple. We have called the input port, PORT B. The bit



16K Ram Expansion Board for the Apple II*

$195.00

- expands your 48K Apple to 64K of programmable memory
- works with Microsoft Z-80 card, Visicalc, LISA ver 2.0 and other software
- eliminates the need for an Applesoft* or Integer Basic ROM Card
- switch selection of RAM or mother board ROM language
- includes installation and use manual
- fully assembled and tested

Visa and MasterCard accepted
Shipping and handling will be added unless the order is accompanied by a check or money order
N.C. residents add 4% sales tax

*Apple II and Applesoft are trademarks of Apple Computer, Inc.

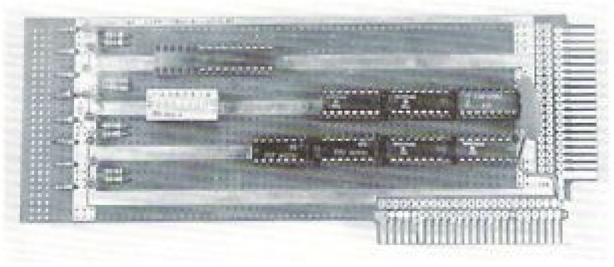
ANDROMEDA
INCORPORATED**
P.O. Box 19144
Greensboro, NC 27410
(919) 852-1482

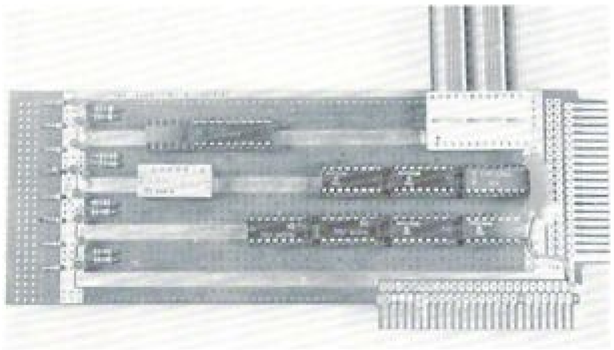
**Formerly Andromeda Computer Systems

values of Port A many be controlled manually by the eight switches on the DIP switch package. If the input pin is connected to ground through the switch. a logic zero results; otherwise you get a logic one. If the input port is to be driven by some other circuit, then the pull-up resistors (2200 ohms) are not necessary. A 74LS244 octal buffer may be substituted for the 81LS97, but they are not pin-for-pin compatible. This completes our description of the input port, and the entire circuit.

Construction

You will need a peripheral card, the integrated circuits, a pin-out diagram for the peripheral card slots (see your Apple manual), soldering equipment, wire-wrap equipment, and several other parts. If you have never wired or built a circuit before, be sure you have someone with experience around who can help you. Two photographs indicate the parts layout that we used. Figure 2 shows the I/O card with the LEDs and switches installed. The pull-up resistors had not yet been wired. Figure 3 shows the I/O card with a ribbon cable DIP jumper connected to a 24 pin socket on the peripheral card. The 24 line cable can handle the I/O port lines and whatever power connections (+ 5V, GND, + 12V, or -12V) that you might want to steal from the Apple. Another cable alternative would be to use an edge connector on the peripheral card and have the cable exit through one of the slots behind the Apple.





I chose to wire-wrap most of the board, and I purchased my wire-wrap kit (WK-2) and my peripheral card (Vector #4609) from Jameco Electronics, 1355 Shoreway Road, Belmont, CA 94002. The remainder of the parts are available from a variety of mail-order houses, Jade Computer Products for example.

The best way to start is to make all of the power connections first. These are not shown in Figure 1, but they are completely described in Table 1. Also, although none are shown in the photographs, it is good practice to install one 0.01 microfarad capacitor between +5V and ground for each two integrated circuits before doing the remainder of the wiring. I installed mine later because I did not have any available on the day that I wired the circuit and took the pictures. Next, wire the lines from the edge connector to the appropriate pins on the wire-wrap sockets. These lines all appear to the left of the circuit diagram in Figure 1, and a table of the pin numbers is given in Table 2. Finally, wire the connections between the pins on the sockets. The procedure just described insures that all the soldering is done at an early stage, when you are less likely to burn through several innocent wires while trying to solder another one. Again, if you have had little experience in wiring, find someone to help you lay out the circuit and get started wiring. Most people are anxious to demonstrate their expertise and be helpful.

Testing and Operation

To test the board first turn off the Apple and install the board in one of the slots, say slot 6 in which case the address of the output port is $ C0E0 (decimal 49376) and the address of the input port is $C0E4 (decimal 49380). (Actually, both ports respond to other addresses also, but that will not be of any concern here.) With the card installed and the Apple running in its monitor, write some number to the output port; that is, enter *C0E0:55 RETURN. The LEDs should display $55. Try some other numbers such as $01, $02, $04, $08, $10, $20, $40, and $80 if you wish to try all the LEDs in turn. To test the input port with the monitor simply use it to examine location $C0E4 by entering *C0E4 RETURN. The number you get should correspond to the switch settings. Try each switch in the logic one position in turn to verify that everything is working properly.

To test the I/O card using BASIC you must PEEK at the input port and POKE data to the output port. For example, a statement

```
10 Y = PEEK(49380)
```

returns the data at the input port as Y. The statement

```
20 POKE 49376, Y
```

will put the contents of Y in the output port, provided Y is not greater than 255. Try running this program:

```
10Y = PEEK(49380)
20 POKE 49376, Y
30 GO TO 10
```

It reads the input port and writes that number to the output port. Thus, the LEDs should follow the input switches. In the above test procedure, we are still assuming that the peripheral I/O card is in slot 6. If you select some other slot, then the addresses used above must be modified.

Concluding Remarks

You should be aware that you cannot use read-modify-write assembly language instructions (e.g., INC, DEC, ASL, LSR, etc.) to reference the output port because it does not include an output register. The best way to reference the output port is with STA, STX, or STY instructions. To demonstrate the other instructions, perform all your read-modify-write instructions on some R/W memory location (RAM), then transfer the contents of this location to the output port. For example this program segment demonstrates the ASL instruction.

```
ASL MEM1 Modify the MEM1 location.
LDA MEM1 Transfer it to the accumulator.
STA PAD Output the data to Port A
```

If you have any questions, please include a self-addressed stamped envelope, and I will be glad to respond.

Table 1. Integrated Circuit Information.

IC NUMBER	DESCRIPTION	QUANTITY	+5V	GND
74LS243	Quadruple Bus Transceiver	2	Pin 14	Pin 7
74LS32	Quadruple 2-Input Or Gates	1	Pin 14	Pin 7
74LS04	Hex Inverter	1	Pin 14	Pin 7
74LS138	3-to-8 Line Decoder	1	Pin 16	Pin 8
74LS75	4-bit Bistable Latch	2	Pin 5	Pin 12
81LS97	Octal Three-State Bus Driver	1	Pin 20	Pin 10

Table 2. Some Pin Numbers for the Apple Bus.

Label (Figure 1)	Pin Number on the Apple Bus
D0	49
D1	48
D2	47
D3	46
D4	45
D5	44
D6	43
D7	42
DEVICE SELECT DS	41
01	38
R/W	18
A0	2
A1	3

©

SBCS

SOFTWARE for MAXIMUM PERFORMANCE WITH YOUR **Apple II**

GENERAL LEDGER
offers you

Virtually complete flexibility in formatting balance sheets and income statements.

- 31-character account names.
- 6 digit numbers.
- 10 levels of subtotals for more detailed income statements and balance sheets.
- up to 9 departments.

A cash journal that automatically calculates the proper off-setting entry and allows a 33-character transaction description.

We are the authorized center for Osborne/McGraw-Hill providing you with business packages that will do everything the Osborne Software will do in addition to many features we have added.

A balance sheet and income statement for the current month, quarter, or any of the previous three quarters.

ACCOUNTS RECEIVABLE allows you to:

- Enter invoices at any time.
- Keep track of invoice amounts, shipping charges, and sales tax (automatically computed).
- Accumulate total payments including progress billing information on each invoice.
- Print reports which list unbilled invoices, unpaid invoices, and paid invoices.
- Obtain an aging analysis of unpaid invoices.
- Assign your own alphanumeric customer code.
- Maintain the date of the last activity for each customer, as well as amounts billed this year and last year.
- Print Customer Statements. (Statements available through SBCS).

Accounts Receivable is available independently or can be integrated with the General Ledger program.

In the final analysis, making your bookkeeping easier is what our software is all about. There is virtually no limit on entries since you may process them as often as you like. These packages will support any printer/interface combination.

General Ledger requires one hundred ten columns.
Accounts Receivable requires one hundred thirty columns.

Suggested Retail:
Individually $180.00
Together $330.00
McGraw-Hill manuals (required for documentation) ..... $20.00 ea.
Available from your local Apple Dealer or contact SBCS.

YOU NEED EXPERIENCE WORKING FOR YOU!
Contact or write:

SMALL BUSINESS COMPUTER SYSTEMS
4140 Greenwood — Lincoln, Nebraska 68504 — (402) 467-1878

List Apple Integer Basic Programs One Page At A Time

Keith Falkner,
Toronto, Canada

The obvious way to examine an unfamiliar program is to type "LIST". In APPLE's INTEGER BASIC, this often gives little or no useful information, because the whole program is listed at great speed, and the moving display can scarcely be read. If the listing could be stopped, this would be no problem; however, only the RESET key stops the listing. Pressing the RESET key is brutal and inelegant, and can cause loss of the program being listed.

This small program in Assembly Language provides a convenient way to list INTEGER BASIC programs without those two problems. It lists one screen-full of the BASIC program, then waits for any key to be pressed. If any key but CTRL-C is pressed, the next screen-full of the program is listed, and so on until the whole BASIC program has been displayed. At any time, CTRL-C can be entered, and the listing ceases, with one screen-full of the BASIC program still visible. This makes it simple to browse an INTEGER BASIC program either quickly or slowly, and stop after any screen-load ("page").

This program does not interfere with BASIC, and as listed here, it occupies a part of memory where it will not likely be damaged. Locations 700-762 ($2BC-$2FA) are approximately the final quarter of the 256-byte keyboard input buffer, and are used only if more than 188 characters are entered as a line of BASIC or in reponse to an INPUT instruction. Either of these is very unusual, and in practice, the program is not over-written.

Users with little experience in machine language can easily enter this program with the Mini-Assembler which is part of APPLE's Monitor, as follows:

```
> CALL - 151 (enter the Machine-language monitor)
*F666G        (enter the Mini-Assembler)
!2BC:LDACA (no need to type spaces or $)
! STAE2       (a space is needed after the !)
! LDACB       (and so on ...)
```

For the "branch" instructions, BCC, BNE, BCS, and BPL, the actual address branched-to is needed. For example:

```
! CMP4D       (the instruction on line 0027)
! BNE2E0      (it branches to SHOWME at $2E0)
```

APPLE suggests using the RESET key to exit the Mini-Assembler, but there is a gentler way:

```
!$FF69G       (type it as shown, with no spaces)
```

The program can be saved on disk via:

```
*BSAVE LISTAPAGE,A700,L63
```

It can be saved onto tape via:

```
*2BC.2FAW (there will be only 1 "beep")
```

At any time, this program can be loaded into memory without disturbing any BASIC program already present. To load it from disk, type:

```
>BLOAD LISTAPAGE
```

To load it from tape, a more complicated sequence is needed:

```
>CALL -151 (to Monitor again)
*2BC.2FAR   (press PLAY before pressing RETURN)
*E003G      (or CTRL-C)
```

In either case, the program is safely hiding in locations 700-762 inclusive, and it can be used in these ways:

To list a BASIC program from the beginning, just type "CALL 700" to see the first page. Press any key except CTRL-C to see more, or press CTRL-C to stop listing after any page.

The program has a second entry-point which is also useful. Type "CALL 708" to resume listing a program after the line most recently listed. For example, to list some lines starting with line 2000, type "LIST 1999", whether or not such a line exists, then type "CALL 708", and successive pages starting with line 2000 will be listed. "CALL 708" can also be used to resume a listing which had been begun by "CALL 700" and stopped by CTRL-C.

Experienced users of machine-language will have noticed that this program is relocatable. In other words, it does not contain any reference to its own absolute address. That in turn means that it can occupy any locations in memory that are not in use for other purposes, and function there without needing any changes. Other locations which can be used to contain this program include, from most convenient to least:

Page 3, locations 768-830 (or nearby) is easiest because neither the APPLE monitor nor BASIC makes use of this space, hence of course, it is the popular place for noise-making routines and various other uses which would conflict with this.

2048-2110. In issue CLR and LOMEM:2110 to prevent BASIC variables from over-writing the routine.

16322-16384. Issue NEW and HIMEM:16322 to prevent a BASIC program from over-writing the routine. Those addresses interfere with High-Resolution graphics, and will be different in a machine with more or less than 16K of memory.

Other locations, such as the gap above the variables and below the program might be tried if none of the above appeals. Experiment at will in this fashion, and remember, "You can't hurt the computer by pressing keys".

This program provides a helpful alternative to the "LIST" command, filling an irritating gap in APPLE's flexible and rapid INTEGER BASIC.

```
LOC  CODE   LINE

0000                .OPT NOSYM
0000        ;    LIST INTEGER BASIC PROGRAM
0000        ;    ONE SCREEN-FULL AT A TIME:
0000        ;
0000                *=700
02BC        ;
02BC        ;    **ENTER HERE TO LIST FROM START
02BC        ;
02BC A5CA           LDA $CA         ;INIT POINTERS TO
02BE 85E2           STA $E2         ;START OF PROGRAM
02C0 A5CB           LDA $CB
02C2 85E3           STA $E3
02C4        ;
02C4        ;    **ENTER HERE TO RESUME LISTING
02C4        ;
02C4 A54C           LDA $4C
02C6 85E6           STA $E6         ;LIST UNTIL
02C8 A54D           LDA $4D         ;HIMEM: HIT
02CA 85E7           STA $E7
02CC 2C10C0 RESUME  BIT $C010       ;RESET KEYBOARD
02CF 2058FC         JSR $FC58       ;CLEAR THE SCREEN
02D2        ;
02D2        ;    SEE IF THERE IS MORE TO LIST.
02D2 A5E3   ANYMOR  LDA $E3
02D4 C54D           CMP $4D         ;ALL DONE?
02D6 9008           BCC SHOWME      ;NO.
02D8 D01B           BNE EXIT        ;YES.
02DA A5E2           LDA $E2         ;MAYBE ...
02DC C54C           CMP $4C         ;FOR SURE?
02DE B015           BCS EXIT        ;YES.
02E0 206DE0 SHOWME  JSR $E06D       ;LIST ONE LINE
02E3        ;
02E3        ;    SEE IF ROOM TO LIST ANOTHER LINE.
02E3 A525           LDA $25         ;CURRENT LINE ON SCREEN
02E5 18             CLC
02E6 6904           ADC #4          ;LEAVE ROOM FOR 4 LINES
02E8 C523           CMP $23         ;ROOM FOR ANOTHER LINE?
02EA 90E6           BCC ANYMOR      ;Y: GO TRY TO LIST MORE
02EC        ;
02EC        ;    SCREEN IS FULL. WAIT FOR A KEY.
02EC AD00C0 WAITKY  LDA $C000       ;SEE WHICH KEY PRESSED.
02EF 10FB           BPL WAITKY      ;NONE. KEEP ON WAITING.
02F1 C983           CMP #$83        ;WAS IT CTRL-C?
02F3 D0D7           BNE RESUME      ;N: DO ANOTHER PAGE.
02F5 2C10C0 EXIT    BIT $C010       ;RESET KEYBOARD.
02F8 4C03E0         JMP $E003       ;BACK TO BASIC.
02FB                .END
```

©

The 25¢ Apple II Real Time Clock

Erann Gat
Oak Ridge, Tennessee

It is interesting to count the number of features of the Apple II which traditionally require boards full of parts to implement, but are done with only one or two inexpensive chips. For instance, the analog to digital conversion for the game paddles would normally cost at least $25, but is done on the Apple with a single inexpensive timer chip. The refresh for the dynamic memory requires no extra parts at all as this is done by the video circuitry.

This philosophy of doing things the easy way makes one wonder at the prices that are being charged for some of the peripheral boards for the Apple, particularly real time clocks. A search for an easier (and hopefully cheaper) way yielded a clock with good accuracy and any feature found on the more expensive boards, including many extra fringe benefits, with a total cost of between 3 to 25 cents depending on how sophisticated you want it to be.

All About Interrupts

Interrupts are something almost every computer hobbyist has heard of, but most of the information about them is rather cryptic. This section will attempt (note that verb) to clarify how interrupts work because they form the basis of the 25 cent clock.

Here is how an interrupt works: on the 6502 microprocessor there are two pins called IRQ and NMI. IRQ stands for Interrupt ReQuest and NMI stands for Non Maskable Interrupt. When either one of these pins is grounded, the processor finishes the machine language instruction it is currently working on, saves the program counter and processor status register onto the stack, (if you don't know what that means it isn't important) and jumps to a program somewhere in memory called an interrupt handling routine or interrupt handler. It then executes the interrupt handler until it encounters a RTI (ReTurn from Interrupt) instruction. It then restores the status register and program counter to their original values and continues executing the main program at the point where the interrupt occurred.

The main program is not affected by an interrupt except that some time is lost during the interrupt and the main program slows down. How much it slows depends on the length of the interrupt handler.

Now suppose that the interrupt handler was a routine that incremented a memory location and returned. This would then be an interrupt counter; i.e. every time an interrupt occurs, the counter is incremented. Now suppose that a pulse was applied to the interrupt line exactly once each second. Voila! A real time clock that tells time in seconds. This is the idea behind the 25 cent clock.

More About Interrupts

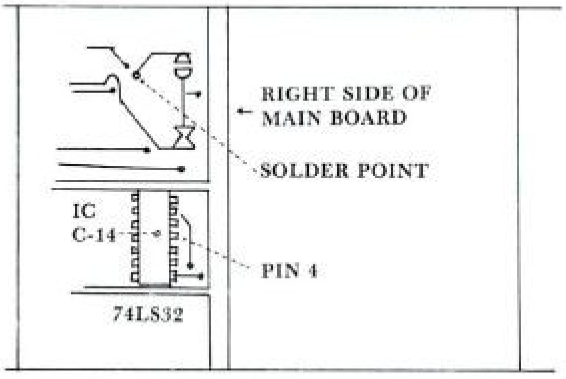
Up until now the 25 cent clock has been discussed in generalities and theories. This section discusses the actual implementation.

First some more facts about interrupts on the 6502: There are two main differences between the IRQ and NMI interrupts. In the 6502 status register there is a flag called interrupt enable. This flag can in effect turn off the IRQ line. If the enable flag is not set, the 6502 will deny Interrupt ReQuests. It will ignore them as if they were not there. On the other hand, NMI cannot be turned off. When a Non Maskable Interrupt occurs, the processor will always act on it and jump to the interrupt handler.

The second difference is that NMI and IRQ have their interrupt handlers at different places in memory. IRQ has another difference in that its interrupt handler is the same routine which handles the BRK instruction. BRK in effect generates a IRQ signal. There is a way to tell IRQ's from BRK's (in fact the Apple monitor does this for you) but this takes up quite a bit of time as well as creating other complications. NMI therefore is more suitable than IRQ for the clock. However, there is no law that says IRQ can't be used.

Next, a signal of known frequency must be found. A time base generator can be used, but at several dollars a piece it would be difficult to stay within the 25¢ budget. An ideal signal can be found in the video circuitry. This signal is the 60 Hz (meaning 60 times each second) pulse which generates the vertical retrace. This signal can be tapped at two locations shown in figure 1. The physical details are discussed in the next section.

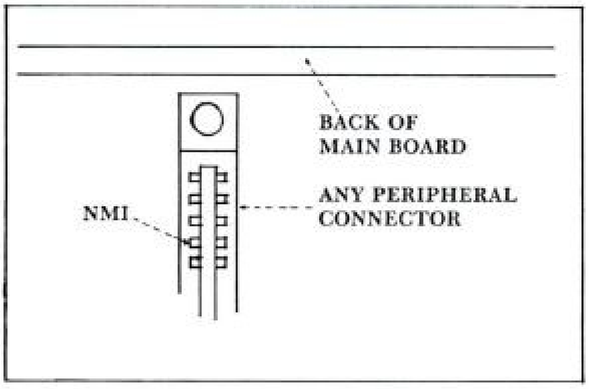
FIGURE 1



The Three Cent Clock

Implementing the clock in its simplest form involves simply connecting the NMI line to a signal source. On the Apple, the NMI line can be accessed from any of the peripheral slots on the rear of the board. The location of the NMI line is shown in figure 2. The connection can be made using a prototype board or by simply inserting a wire between the metal contact and the plastic housing of the connector.

FIGURE 2



The 60 Hz signal can be accessed in the two locations shown in figure 1. The first place is a small solder filled hole in the board. A wire may be soldered in the hole, or a wire wrap pin may be attached and connected to the NMI line via an alligator clip to make the clock removable. **NOTE; This may void your warranty. Check with your dealer!**

The other connection point does not involve soldering. To make the connection, carefully remove the IC at location C-14. The row and column numbers are marked on the board itself. Then insert a piece of very thin (wire wrap) wire into pin 4 of the socket. (See figure 1.) Now carefully reinsert the IC making sure it is oriented correctly and all the pins are securely seated in the socket.

Before this connection is made an interrupt driver must be entered into memory. If this is not done, the system will crash and RESET will have no effect until the connection is broken.

To get the three cent clock off to a flying start, enter the short program in listing 1. This can be done in the monitor or the mini-assembler. When the program is in memory, connect the interrupt line and watch the upper left hand corner of the screen. If everything was done correctly, the first character on the screen should start changing rapidly. What is happening is that sixty times a second the video circuitry generates a signal which is now being used to generate an interrupt. When an interrupt occurs, the processor starts executing the interrupt handler which is located at 3FB hexadecimal on an Apple. Usually the interrupt handler starts with a jump instruction since there are only five bytes of usable memory at

LISTING 1
```
*3FBL
03FB-     EE 00 04      INC     $0400
03FE-     40            RTI
03FF-     00            BRK
```

3FB, but since this program is so short it can be entered directly at 3FB. The interrupt handler that is now in memory simply increments a memory location and returns to the main program. This is a real time clock. It tells time in sixtieths of a second. Granted, it isn't very useful as it is now, but that will be fixed in a moment.

Now incrementing a memory location on the screen isn't very exciting, but try hitting a few keys. Surprise! They still work. In fact, everything works. Try dumping out some memory or printing something in basic. Everything will work normally and the first character on the screen will go right on counting. WARNING: the disk will **NOT** work. Neither will the tape. This is because the interrupts slow down the main program enough to upset the precise timing required by the disk and tape routines. Having the interrupt connected will also make the bell tone sound peculiar.

To make the clock more useful, enter the three programs in listing 2. The first program is simply a jump instruction to the second program which is a clock routine to drive an hour-minute-second clock. The third program is a basic routine which sets the clock and outputs the time of day. The programs are thoroughly documented so they won't be discussed here.

LISTING 2
PROGRAM #1
```
*3FBL
03FB-     4C 00 03      JMP     $0300
03FE-     00            BRK
03FF-     00            BRK
```

Making It Better or When Is An NMI Really An IRQ?

It should be clear by now that the power of the clock lies in the interrupt driver program, but there are some hardware enhancements that can be made. These extra features will roll the price up to a respectable 25 cents (more or less).

The first add-on is a sophicitcated piece of hardware called a switch. This is used to make easier the task of turning the interrupts on and off. The switch is installed so that it breaks the connection from the 60 Hz signal. Personal experience has shown that flipping a switch makes a more dignified display than pulling a wire in and out.

The second modification is a bit more complicated. (Seriously.) This modification allows the computer to control the interrupts via one of the annunciator outputs on the game I/O connector. The

LISTING 2
PROGRAM #2

```
*300LL                CLOCK
0300-   85 05          STA   $05          SAVE A AND X
0302-   86 06          STX   $06
0304-   A9 3C          LDA   #$3C          A=60 DECIMAL    X=0
0306-   A2 00          LDX   #$00
0308-   E6 04          INC   $04           COUNT 1/60 SECOND
030A-   C5 04          CMP   $04            FULL SECOND YET?
030C-   D0 22          BNE   $0330           IF NO THEN RESTORE REGISTERS & RETURN
030E-   86 04          STX   $04           RESET 1/60 SECONDS
0310-   E6 03          INC   $03           COUNT 1 SECOND
0312-   C5 03          CMP   $03            1 MINUTE YET?
0314-   D0 1A          BNE   $0330
0316-   86 03          STX   $03
0318-   E6 02          INC   $02           MINUTES
031A-   C5 02          CMP   $02
031C-   D0 12          BNE   $0330
031E-   86 02          STX   $02
0320-   A9 0D          LDA   #$0D          SET A=# HOURS IN 1 DAY PLUS 1
0322-   E6 01          INC   $01           HOURS
0324-   C5 01          CMP   $01           FULL DAY?
0326-   D0 08          BNE   $0330
0328-   E8             INX                 IF YES SET HOURS TO 1
0329-   86 01          STX   $01
032B-   A5 05          LDA   $05           RESTORE REGISTERS
032D-   A6 06          LDX   $06
032F-   40             RTI
0330-   A5 05          LDA   $05           RESTORE THEM HERE TOO
0332-   A6 06          LDX   $06
0334-   40             RTI
0335-   00             BRK
0336-   00             BRK
0337-   00             BRK
0338-   00             BRK
0339-   00             BRK
```

LISTING 2
PROGRAM #3

```
>LIST CLOCK DRIVER
  5 PRINT CHR$(4);"BLOAD CLOCK"
  7 POKE 1020,0: POKE 1021,3: REM  SET INTERRUPT
    VECTOR
 10 INPUT "INPUT TIME -->",H,M,S
 15 REM  SET CLOCK
 20 POKE 1,H
 30 POKE 2,M
 40 POKE 3,S
 43 POKE 4,0
 45 A= PEEK (-16296): REM  TURN CLOCK ON
 47 INPUT "12 OR 24 HOUR CLOCK",A: POKE 801,A+1
 48 REM  SEE LISTING   FOR EXPLAINATION OF LINE 47
 50 CALL -936: REM  CLEAR SCREEN
 60 VTAB 10: PRINT "
 61 REM  ERASE OLD TIME
 70 VTAB 10: TAB 10
 75 REM  DISPLAY CURRENT TIME
 80 PRINT PEEK (1);":";
 81 REM  HOURS
 90 IF PEEK (2)<10 THEN PRINT "0";: PRINT PEEK (2),
 91 REM  MINUTES
100 PRINT PEEK (3), PEEK (4): GOTO 60
110 REM  SECONDS AND 1/60 SECONDS
```

NEW FOR APPLE 2 & APPLE 2 Plus

PLANETOIDS

A REAL TIME HI-RESOLUTION GRAPHICS GAME WITH SUPER SOUND EFFECTS. THREE SKILL LEVELS - NOVICE, AVERAGE ("Arcade" difficulty") and EXPERT WHERE ASTEROIDS ARE ACTUALLY ATTRACTED TO YOUR SHIP!

PLANETOIDS
Formerly "ASTEROID" a T M of Atari

Now Available

32K TAPE - $14.95
32K DISK - $19.95

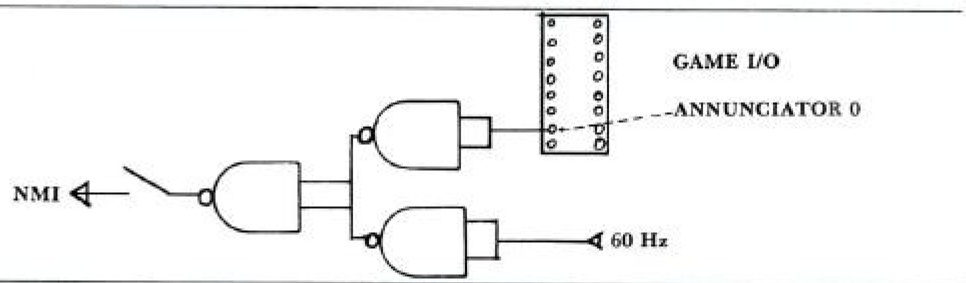
From Your Local Computer Store Or Send Order to:

© Copyright 1980
Adventure INTERNATIONAL
A DIVISION OF SCOTT ADAMS, INC

BOX 3435
LONGWOOD FLA 32750
305-862-6917

DEALER INQUIRIES INVITED

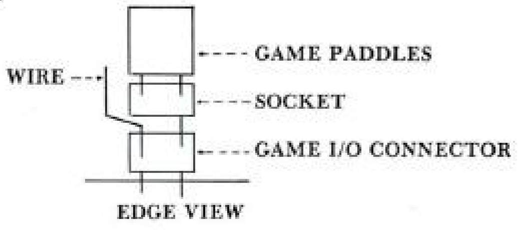
FIGURE 3



only extra part required is a 7400 or 74LS00 nand gate. It is wired according to figure 3 using a prototype board, an off-board wire wrap socket, or the breadboard area on the Apple board. Even the revision 1 boards have room for two IC's in the right hand corner under the keyboard. NOTE: To wire the modification in this way requires removal of the Apple board and will probably void your warranty. Check with your local dealer.

The connection to the game I/O connector is made using a piece of stiff wire such as the lead of a small resistor. This wire is inserted into the connector and bent as shown in figure 4. A 16 pin IC socket with one pin clipped to accommodate the wire is inserted over that and the game paddles are plugged into that socket. Many connections can be made to the game connector in this manner without having to clip pins off of the game paddles.

FIGURE 4



The Disadvantages

Unfortunately, every silver lining comes equipped with a cloud and the 25 cent clock is no exception. The main problem is that the disk and tape will not work, as well as other programs which involve precise timing. The interrupts must be disabled, either manually or under program control, while such programs are running.

Another hitch is in the computer control circuit itself. When an Apple is turned on, the annunciator outputs are high (logic 1) so this has been made to disable the interrupts. An autostart rom however, turns all the annunciators to logic 0. Before this happens all the annunciators are still at logic 1 for a few milliseconds so inverting the signal from the annunciator will still leave the interrupts enabled for enough time to cause an interrupt and a system crash. Therefore, the interrupts must be disabled manually upon power up with an autostart rom.

Another problem is that the bell tone sounds raspy. This isn't serious, but it can get on your nerves after a while. It doesn't make a good way to check if interrupts are enabled.

The final problem is that the clock seems to lose about ten seconds each hour. This can be remedied by adding ten seconds to the seconds counter each hour.

Fringe Benefits

The 25 cent clock is remarkably user proof. The NMI line doesn't require debouncing, and resetting the comupter doesn't interfere with its operation either (unless the reset key is held down for a long time).

The two main dangers of system crashes are working on the interrupt handler while interrupts are enabled, and not saving registers. THIS IS IMPORTANT!!! You must save each register you intend to modify. If you do not you will get very mysterious results. You can save registers in memory or you can push them onto the stack. There is also a routine to save and restore all registers in the monitor.

Once these restrictions have been met, the 25 cent clock opens a vast new horizon of features that would cost tens of dollars if bought from vendors. The price you pay is speed. The longer the interrupt routine, the slower the computer runs. This is not a severe handicap. The clock routine does not slow the computer down enough to be perceived, even when the interrupts are switched on and off for comparison. In order to slow the computer down by even one percent it requires a one hundred instruction routine.

Some things that can be done include:

Control Of Computer Speed Using Game Paddles: have the interrupt driver pause according to the position of a game paddle to give control of listing speed, how fast a program runs, etc.

Keyboard Buffering: have the interrupt routine sample the keyboard and store any keypresses in a buffer to give storage of multiple keypresses while something else is going on.

Mixing Display Modes: sixty times a second switch to another display mode to mix text and graphics, or mix two graphics modes for extra colors.

The possibilities are endless. You can even run two programs at once using the interrupt. The twenty-five cent Apple II real time clock is a lot more than just a clock, it's a cheap way of doing a lot of expensive things, right in line with Apple tradition.

GLOSSARY

INCREMENT- to add 1 to a counter
INTERRUPT HANDLER- a machine language program which is executed whenever an interrupt occurs
INTERRUPT VECTOR- the address of the interrupt handler routine
IRQ- Interrupt ReQuest; an interrupt line which can be disabled under program control
NMI- Non Makable Interrupt; interrupt line which cannot be disabled
REAL TIME CLOCK- a device which provides a computer with information about the time without disrupting the computer's normal functions ©

Apple Monitor Extender

APPLE II 16K, CASSETTE

This utility program works in complete harmony with the Apple monitor to extend your computer's capability and help you use the full power of machine language programming.

Screen display shows memory in HEX, ASCII or BINARY. Move data anywhere in memory without regard to direction or overlapping and read or write any sector on disk. Insertions may be in HEX or ASCII so you can easily format high speed text displays without conversions.

Study, modify or disassemble any program, complete with labels. Several programs may be combined, and the entire disassembled text file stored on disk/tape for later assembly.

The slow listing feature steps through listings with ease.

© Copyright 1980 Glenn R. Sogge, All Rights Reserved.

IMAGE COMPUTER PRODUCTS™
615 Academy Drive
Northbrook, IL 60062
312/564-5060

HUNTINGTON COMPUTING

PROGRAMS LISTED BELOW ARE ON DISK:

Apple Galaxian	$24.95 now	**$22.99**
EPYX (Automated Simulations) SPECIAL — While they last: Ryn/Morloc/Rigel all three		**$39.95**
Galaxy Wars	$24.95 now	**$22.99**
Sorcerer/Doom Cavern	$20.00 now	**$17.99**
Hyper Head-On	$24.95 now	**$21.20**
Galactic Empire	$24.95 now	**$21.20**
Galactic Trader	$24.95 now	**$21.20**
Galactic Revolution	$24.95 now	**$21.20**
Golden Mountain	$19.95 now	**$16.95**
Tank Command	$14.95 now	**$12.49**
DB Master	$189.00 now	**$169.00**
Bloody Murder	$19.95 now	**$17.99**
Tranquility Base	$24.95 now	**$22.99**
Aristotle's Apple	$34.99 now	**$29.99**
Micro Memo	$39.95 now	**$35.99**
The Data Factory	$150.00 now	**$134.00**
Mission Asteroid	$19.95 now	**$17.99**
Vinyl holders for 20 disks in beautiful deluxe padded binder		**$13.99**
Skybombers II	$19.95 now	**$17.99**
Apple-Doc	$24.95 now	**$21.99**
Programmers Utility	$16.95 now	**$15.00**
DOS Tool Kit	$75.00 now	**$69.00**
Microsoft Adventure	$29.95 now	**$26.09**
Olympic Decathlon	$29.95 now	**$26.09**
Compu-Math: Arithmetic	$49.95 now	**$44.95**
Apple Plot	$75.00 now	**$69.00**
Memorex disks - Ten for		**$37.99**
College Boards (Krell)	$79.95 now	**$69.95**
Time Traveler	$24.95 now	**$22.99**
Sword of Zedek	$24.95 now	**$22.99**
All Time Super Star Baseball	$14.95 now	**$13.19**
CRAE 2.0	$24.95 now	**$21.99**
MCAT 2.0	$19.95 now	**$17.99**
The Tarturian	$24.95 now	**$21.99**
TI PROGRAMMER — Hexadecimal/Octal Calculator Retail $65.00	now	**$59.99**
MPI 88G Printer		**$699.00**

We sell the complete line of Bell and Howell Apple Computers — Write or call for quotes.

The Prisoner	$29.95 now	**$26.99**
The Wizard & The Princess	$32.95 now	**$28.99**
Comps-Spell	$39.95 now	**$34.99**
Computer Ambush	$59.95 now	**$52.99**
Computer Bismark	$59.95 now	**$52.95**
Computer Napoleonics	$59.95 now	**$52.99**
Computer Quarterback	$39.95 now	**$35.99**
Flight Simulator (disk)	$34.95 now	**$27.20**
Star Cruiser	$24.95 now	**$20.99**
Space Album	$39.95 now	**$34.50**
Odyssey	$29.95 now	**$25.50**
Both Barrels	$24.95 now	**$20.99**
Modifiable Database	$79.50 now	**$64.99**
Micro League Baseball	$24.95 now	**$22.99**
Sargon II	$34.95 now	**$29.70**
Program Line Editor	$40.00 now	**$34.99**
Z-80 Softcard with CP/M	$349.00 now	**$299.00**
Videx 80-Col. Board	$350.00 now	**$299.00**
EZ Draw	$34.95 now	**$29.99**
Head On	$24.95 now	**$20.99**
3D Graphics	$39.95 now	**$35.99**
Compu-Math I or II	$40.00 now	**$34.00**
Apple World	$59.95 now	**$52.99**
ABM (Muse)	$24.95 now	**$22.49**
Data Plot (Muse)	$59.95 now	**$52.99**
Computer Conflict	$39.95 now	**$35.99**
Computer Air Combat	$59.95 now	**$52.99**
Terrorist	$29.95 now	**$25.45**
The Temple of Apshai	$29.95 now	**$25.45**
Morloc's Tower	$19.95 now	**$16.96**
Super-Text II	$150.00 now	**$129.93**
Magic Window	$99.95 now	**$85.00**
Desktop Plan	$99.95 now	**$85.00**
CCA Data Management	$99.95 now	**$85.00**
Visicalc	$150.00 now	**$119.00**
Acani: Fortune-telling	$15.00 now	**$8.99**
Hornblescope	$15.00 now	**$11.93**
Cyber Strike	$39.95 now	**$36.99**
HI-RES Football (On-Line)	$39.95 now	**$36.99**
Fastgammon	$24.95 now	**$21.20**
The Voice	$39.95 now	**$35.99**
Hellfire Warrior	$29.95 now	**$25.45**
Beneath Apple Manor	$20.00 now	**$17.45**
Astro Apple	$20.00 now	**$17.45**
Akalabeth	$34.95 now	**$29.99**
ANDROMEDA 16K RAM Expansion Board for Apple II Retail is $195 — Our price		**$184.99**
NEC 12" Green/Black	$260 now	**$219.00**
Centronics 730 Printer	$795 now	**$649.00**
Centronics 737 Printer	$995 now	**$799.00**
ABT Pad for Apple	$125 now	**$109.00**
ROM - w/filter	$200 now	**$179.00**
M&R 80-col. Video Card	$375 now	**$325.00**
Versawriter	$249 now	**$219.00**

apple SOFTWARE

SPECIAL up to 40% OFF LIST

FASTEST SERVICE

We take MasterCard or VISA (include card # and expiration date). California residents add 6% tax. Include $2.00 for postage. $5.00 Foreign. Mail to

HUNTINGTON COMPUTING, Dept. COM-2
2020 Charles, Corcoran, CA 93212
24 hour order service. Call
(209) 992-5411. Send for free catalog

Mountain Computer can now EXPAND Your Apple II Peripheral Capacity

EXPANSION CHASSIS

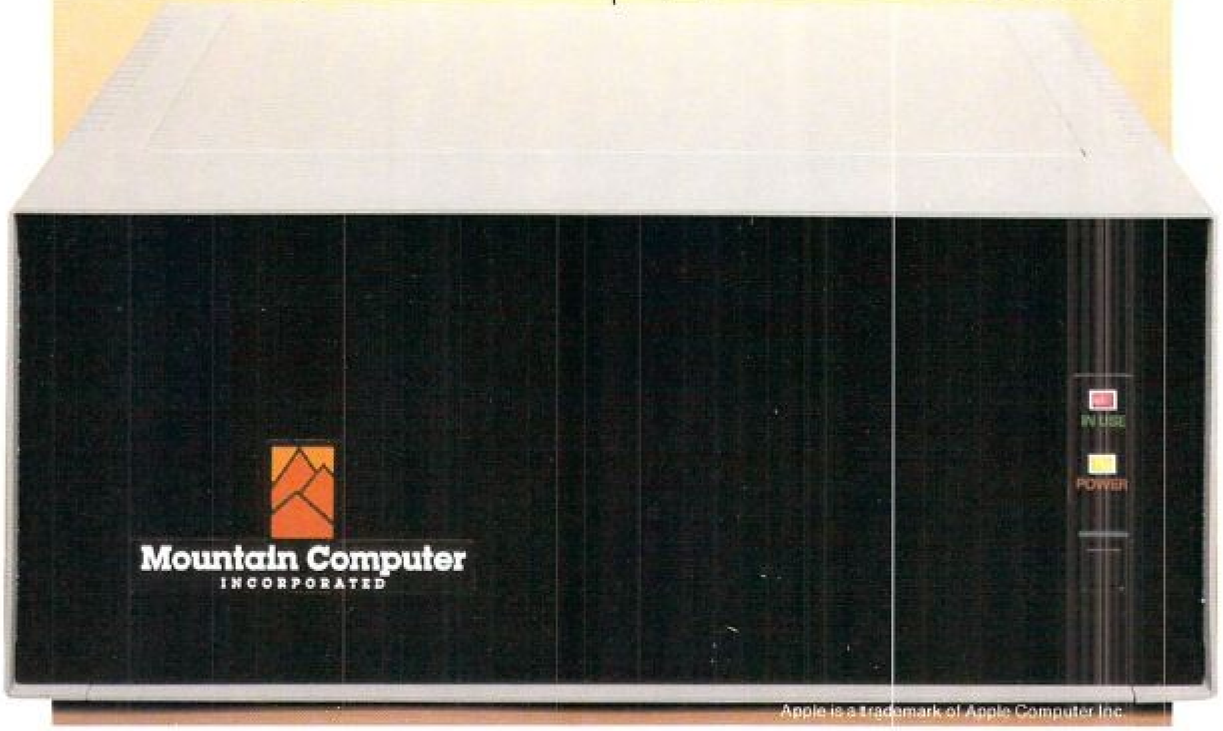
Quality You Expect

Eight more slots for your Apple! Now you can bank-select eight more peripheral slots with immediate or deferred software commands—like having up to 15 peripheral cards "on line"—or use the Select/Deselect switch mounted on the front panel. Expansion Chassis' heavy-duty power supply is primarily for peripherals, without the heavy demand of motherboard support chips required in your Apple. This means much more power is available for peripherals than in your Apple itself! If you've run out of room in your Apple—Expansion Chassis is your answer. Drop by your Apple dealer for a demonstration, or contact Mountain Computer for the location of the dealer nearest you.

Performance You Demand

- Eight mirror image I/O slots of the Apple
- Fully buffered, bi-directional data lines
- Apple II compatible interface card
- Dual selection capability; hardware or software
- Immediate or deferred selection in software mode
- From BASIC, a single POKE command turns the chassis ON or OFF
- Compatible with all software
- Dedicated power supply with approved power transformer

Mountain Computer
INCORPORATED
300 Harvey West Blvd., Santa Cruz, CA 95060
(408) 429-8600 TWX 910 598-4504



WHY THE MICROSOFT RAMCARD™ MAKES OUR SOFTCARD™ AN EVEN BETTER IDEA.

Memory — you never seem to have quite enough of it.

But if you're one of the thousands of Apple owners using the SoftCard, there's an economical new way to expand your memory dramatically.

16K ON A PLUG-IN CARD.

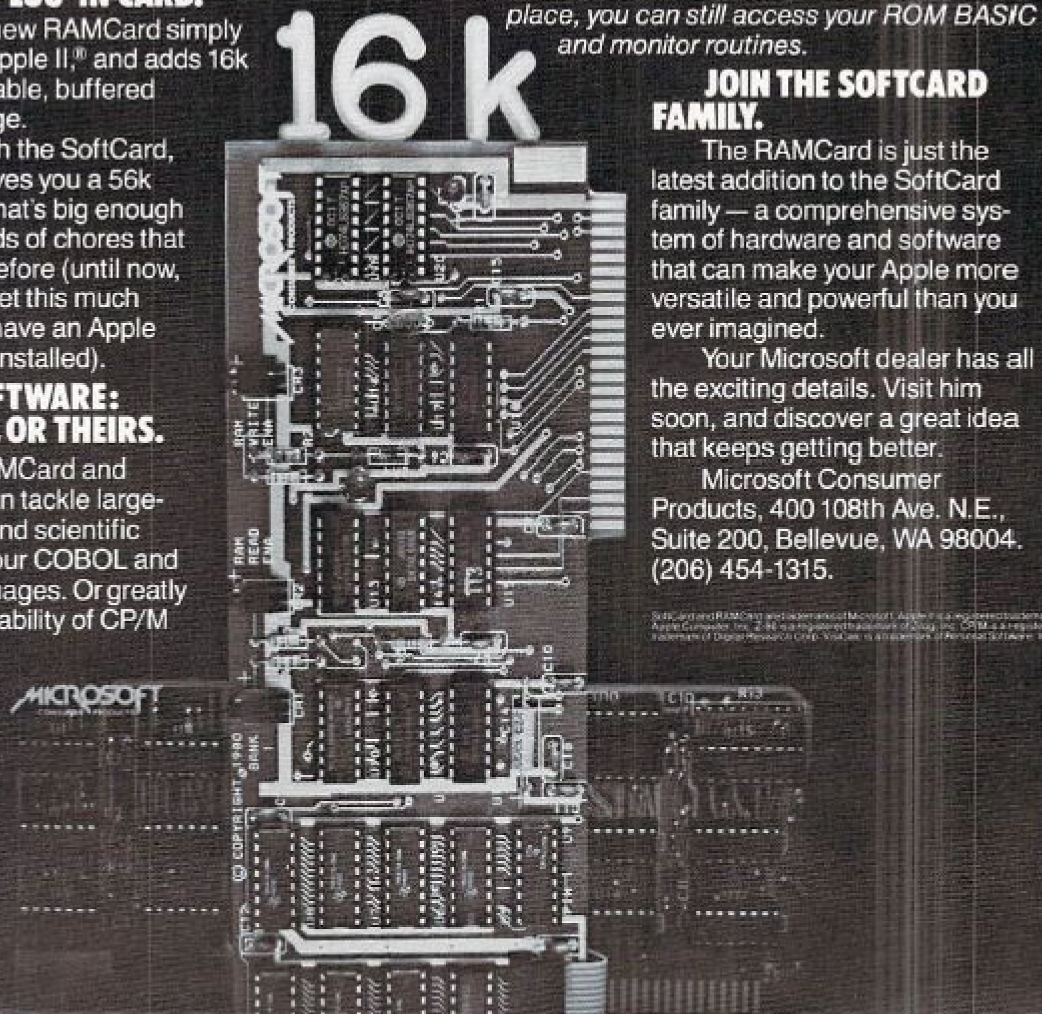
Microsoft's new RAMCard simply plugs into your Apple II,® and adds 16k bytes of dependable, buffered read/write storage.

Together with the SoftCard, the RAMCard gives you a 56k CP/M® system that's big enough to take on all kinds of chores that would never fit before (until now, the only way to get this much memory was to have an Apple Language Card installed).

GREAT SOFTWARE: YOURS, OURS, OR THEIRS.

With the RAMCard and SoftCard, you can tackle large-scale business and scientific computing with our COBOL and FORTRAN languages. Or greatly increase the capability of CP/M applications like the Peachtree Software accounting systems. VisiCalc™ and other Apple software packages can take advantage of RAMCard too.

And RAMCard gives you the extra capacity to develop advanced programs of your own, using the SoftCard and CP/M. *Even with the RAMCard in place, you can still access your ROM BASIC and monitor routines.*



JOIN THE SOFTCARD FAMILY.

The RAMCard is just the latest addition to the SoftCard family — a comprehensive system of hardware and software that can make your Apple more versatile and powerful than you ever imagined.

Your Microsoft dealer has all the exciting details. Visit him soon, and discover a great idea that keeps getting better.

Microsoft Consumer Products, 400 108th Ave. N.E., Suite 200, Bellevue, WA 98004. (206) 454-1315.

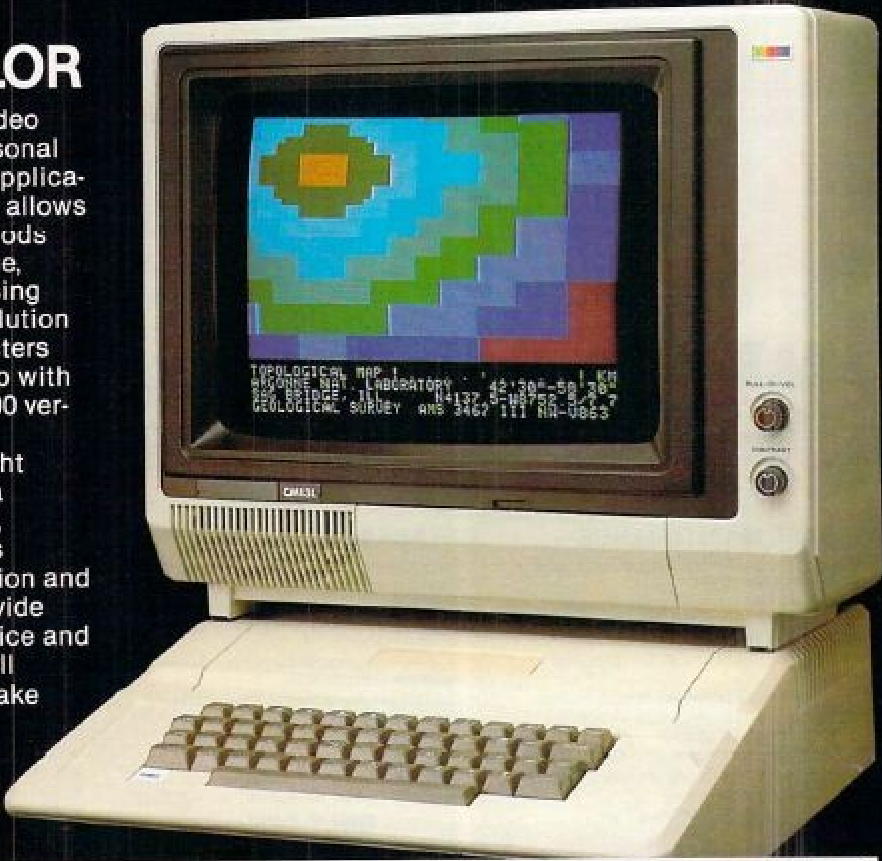
MICROSOFT

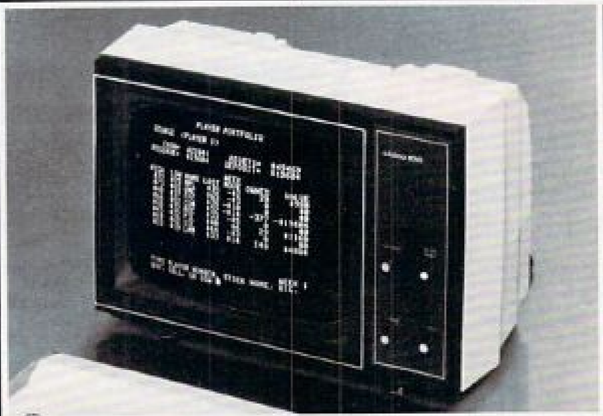
AMDEK Professional Color & B/W Monitors

NEW 13" COLOR

The Amdek 13" Color Video Monitor is ideal for all personal and business computing applications. Bright jitter-free text allows viewing over extended periods without causing eye fatigue, especially in word processing applications. The low resolution display provides 40 characters wide by 24 characters deep with 260 horizontal lines and 300 vertical lines.

This compact, lightweight monitor (only 25 lbs.) has a molded-in carrying handle, making it easy to move. Its industrial grade construction and shatter resistant case provide reliable service in both office and industrial environments. All front mounted controls make it easily adjustable at a glance without the loss of a single keystroke.





VIDEO 100 12" B/W MONITOR

This highly reliable 12" black and white monitor features a 12MHz band width and 80 character by 24 line display. Plug-in compatibility with Apple, Atari, Radio Shack, O.S.I., Micro-Term and Exidy make this the perfect text display for almost any system.

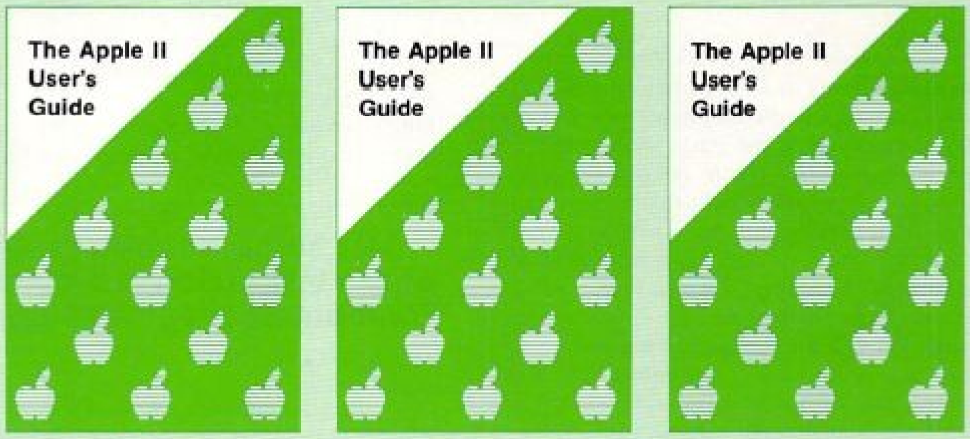
VIDEO 100-80 12" B/W MONITOR

The model 80 features an industrial grade metal cabinet with built-in disk mounting capability and space for an 11" x 14" PC board for custom designed electronics. Front mounted controls include power, contrast, horizontal hold, vertical hold, and brightness.

Amdek Corp. • 2420 E. Oakton Street, Suite "E" • Arlington Heights, IL 60005 • (312) 364-1180 TLX: 25-4786

Amdek Corp. the new name for **Leedex**... Expanding for the future, growing with success.

Osborne Polishes THE APPLE


The Apple II User's Guide
The Apple II User's Guide
The Apple II User's Guide

and Publishes THE GUIDE

The Apple II User's Guide by Lon Poole, Martin McNiff, and Steven Cook #46-2, $15. ☐

This Guide is the key to unlocking the full power of your Apple II or Apple II plus computer. The Apple II User's Guide brings together in one place a wealth of information for Apple computer users. It will tell you more about your Apple than any other single source.

Topics include: **Applesoft and Integer BASIC programming** - especially how to make the best use of Apple's sound, color and graphics capabilities. The book presents a thorough description of every BASIC statement, command and function. **Machine level programming** - although not a machine language programming guide, this book covers the Machine Language Monitor in detail. **Hardware features** - the disk drive and printer are covered in separate chapters. **Advanced programming** - special sections describe high resolution graphics techniques and other advanced applications.

This book will save you both time and effort. No longer will you have to search endlessly for useful information. It's all here, in the Apple II User's Guide, thoughtfully organized and easy to use.

*Apple is a trademark of the Apple Computer Corporation.

More Osborne Books for Apple Users

6502 Assembly Language Programming
by Lance Leventhal
Here's the book that shows you how to program the Apple in assembly language. "With its numerous reference tables, explanations and descriptions, this book is a major accomplishment. It's by far the most complete 6502 book to appear . . . Make space on your bookshelf. You'll want to keep this one handy." *COMPUTE*
Book #27-6 $16.99, 606 pp. ☐

Some Common BASIC Programs
by L. Poole and M. Borchers
Written in a subset of standard BASIC that is compatible with many microcomputers, this book contains 76 useful and easy-to-use programs. The book explains how each program works, shows sample runs, and has program listings with remarks. "If you want to do mathematics or statistical programming on your computer, this book is a must." *ON COMPUTING* Magazine
Book #06-3, $14.99, 200 pp. ☐

Practical Basic Programs
edited by Lon Poole
Here's a collection of 40 programs you can easily key in and use on most microcomputers. *Practical Basic Programs* is especially useful in small business applications. It solves problems in finance, management, and statistics. The book contains sample runs, practical problems, BASIC source listings, and an easy-to-follow narrative to help you realize the potential uses of each program.
Book #38-1, $15.99, 200 pp. ☐

An Introduction to Microcomputers Volume I - Basic Concepts Second Edition
by Adam Osborne
The world's best selling textbook on microcomputers develops a detailed picture of what a microcomputer can do, how it does what it does, and how it can be used in any practical environment. This is the most comprehensive and up-to-date introduction to microprocessor systems available anywhere.
Book #34-9, $12.99, 320 pp. ☐

- -

Make check payable to: OSBORNE/McGraw-Hill
630 Bancroft Way, Berkeley, CA 94710 **Dept. G4**

Phone Orders: (415) 548-2805 or Toll Free 800-227-2895

Name ______________________________

Address ______________________________

City/State/Zip ______________________________

Plus ☐ .75/item 4th class ☐ $1.25/item UPS ☐ $2.50/item Air Mail ☐ $4.00/item Overseas (California residents add applicable tax.)

Total Amount Enclosed $ __________ or charge my ☐ VISA ☐ Mastercharge Card # __________ Exp. Date ______

Signature ______________________________ ☐ Please send me your free catalog.

GIFTS FROM PROMETHEUS

NEW for APPLE II(1)

Advanced products at Down-to-Earth PRICES

DUAL SERIAL CARD-1. All functions of two independent serial cards on one board – plus more. Provides Apple II(1) users with two simultaneous asynchronous serial channels. DSC-1 appears as two separate logical serial cards to the Apple II(1). Fully software compatible with all Apple II's(1) and Microsoft software. 2716 EPROM used for easy driver modification. In addition, the second serial channel supports these extra features: • Strapable DCE/DTE. • Secondary RS-232 handshake functions (DCD, DTR, DSR) • Strapable logical slot location and hardware looks like it's another slot. **Order: PP-DSC-1 @ $189.00 each.**

DUAL SERIAL/PARALLEL CARD-1. As with the Dual Serial Card-1 (above), the Dual Serial/Parallel-1 provides Apple users with simultaneous use of one full serial port and one parallel port. Both ports strapable for any Apple peripheral slot. Fully compatible with all Apple and Softcard (CP/M) software. Drivers for serial interface and Centronics parallel interface are contained in modifiable industry standard EPROM. Serial port is configured like DSC-1. **Order: PP-DSP-1 @ $189.00 each.**

MEMORY EXPANSION MODULE-1. Expand your Apple II(1) to a full 64K RAM system with highest quality 16K dynamic RAMS. Fully buffered to provide reliable operation – even with fully loaded Apples. Total compatibility with Microsoft Softcard(2)/CP/M(3) and all Apple software. Supplied complete with installation instructions and test program. **Order: PP-MEM-1 @ $149.50 each.**

PASCAL MT+ MEM-1. A modular native Z-80 code Pascal compiler. ISO standard. Includes a MEM-1 (detailed above). Have a full Pascal compiler at half the Apple price. Requires a Microsoft Softcard(2) and two disk drives. Note that the Pascal MT+ alone is normally $250.00. **Order: PP-PMT-1 @ $299.00 each.**

16K RAM ADD-ON KITS. For TRS-80 and Apple II(1). Eight (8) full-specification industry standard 16K RAMS. These are not seconds. **Order: PP-RAK-16 @ $29.00 each.**

All Prometheus boards are completely tested and burned-in prior to shipment. One-year warranty covers parts and labor. **MONEY BACK GUARANTEE:** Order now without risk. Boards may be returned, in good condition, after up to 10 days of trial for a full and prompt cash refund. California: add 6% tax; BART Counties, 6½%. Orders under $150, add $3.00 postage, for handling charge. Freight allowed on orders over $150. *Dealer inquiries invited. Quantity discounts available.*

VISA master charge

PROMETHEUS

PROMETHEUS PRODUCTS INCORPORATED

4509 Thompson Ct. • Fremont, CA 94538 • (415) 791-0266

Registered Trade Marks: (1) Apple Computers, Inc. (2) Microsoft Consumer Products (3) Digital Research, Inc.

Clearing The Apple II Low-Resolution Graphics Screen

Sherm Ostrowsky

Many applications require rapidly clearing the low-resolution graphics screen to black (COLOR = 0) or to some other color. In the latter case the process might be more accurately described as "back-grounding". Either way, this apparently simple operation can be done by several different methods. Each method will produce a distinctly different visual effect while in operation, although the end result will be the same. By doing the experiments to be described below, the experienced programmer can learn how to use the method best suited to his immediate purpose, and the novice programmer can learn some useful facts about the operation of the Apple low-resolution graphics. So go ahead and do the experiments on your Apple; you can't hurt it by pushing the keys (even the wrong keys), and you can learn a lot.

First of all, in order to see the effect of any kind of screen-clearing method it is best to begin with a screen that is loaded with colors and forms. You may do this in any way that pleases you; I have been using the following subroutine in Applesoft:

```
1000 GR
1010 FOR I = 0 TO 39
1020 FOR J = 0 TO 39
1030 COLOR = 1 + INT(15*RND(1))
1040 PLOT J, I
1050 NEXT J, I
1060 FOR PAUSE = 0 TO 2000: NEXT PAUSE
1070 RETURN
```

Notice that this subroutine colors-in the so-called "mixed screen" — the top 40 lines, but not the bottom part reserved for text. If you wish to use, and color-in, the whole screen (48 graphics lines), then the first two lines of the Applesoft subroutine can be amended to:

```
1000 POKE -16302,0 : POKE -16304,0
1010 FOR I = 0 TO 47
```

etc. The line of POKEs turns on the "soft switches" governing the full-screen lo-res graphics (see pages 12-13 in the new Apple II reference manual).

Now that the screen is colored, let's clear it. The first method which is likely to occur to the average programmer is to write a couple of lines in Applesoft. Suppose you want to clear the screen to a particular background color, say C (C = 0 to 15). A program to do this for a mixed screen might look like this:

```
10 GOSUB 1000 : REM PAINT THE SCREEN
20 COLOR = C
30 FOR I = 0 TO 39
40 VLIN 39,0 AT I
50 NEXT I
60 END
```

Try it. The screen clears rather ponderously, like a stage curtain rolling across from left to right. If you want the curtain to move from right to left, just change line 30 to

```
30 FOR I = 39 TO 0 STEP -1
```

If you want it to operate on whole-screen graphics, line 40 should be altered to

```
40 VLIN 47,0 AT I
```

This method works fine, if you don't mind the relatively slow speed of the clearing operation. In fact, for some special effects it might even be preferred. Notice how you can control the direction of motion of the apparently rolling curtain. As an "exercise for the student", consider how you might change lines 30 and 40 so as to cause the curtain to appear to be rising upwards. That can be a rather pretty effect, especially if you don't just leave a blank screen but instead "paint-in" a scene of some kind to coincide with the rising of the curtain (i.e., one horizontal line at a time, from bottom to top); it can look like a real stage curtain rising to reveal a scene already in place.

But what if you are not satisfied with the relatively slow speed with which an Applesoft program can clear the screen? If you don't mind being restricted to just a basic black clear, there are some dandy machine-language subroutines in the Apple's built-in ROM Monitor which are a lot faster. For mixed-screen graphics, try this little program:

```
10 GOSUB 1000 : REM PAINT SCREEN
20 CALL -1994
30 END
```

That's not only a heck of a lot faster, but pretty simple to use, too! If you're doing full-screen graphics, replace line 20 with

```
20 CALL -1998
```

ON-LINE systems **IS PROUD TO ANNOUNCE A NEW ADDITION TO OUR FAMILY!**

SuperScribe

(Formerly SuperScript)

SuperScribe is the most powerful and easiest to use Word Processor available for your Apple II or II Plus computer. Besides leaping tall buildings in a single bound it:

- Gives true upper/lower case text on your screen with no additional hardware whatsoever.
- Works with documents larger than the amount of memory in your Apple - transparently to you!
- Edit not only letters but also any text or binary file, or even basic programs!
- Automatically generates up to 4 separate indices for your document!
- Save typing time through a unique ability to designate specified keys as commonly used words, phrases or even commands!
- Globally search for or replace character strings.
- SuperScribe has a built-in instruction capability such that if you forget how to use a command and the manual is not close by - you may simply ask SuperScribe!
- Supports multiple disk drives!
- Will support alternate character sets.
- Produces form letters using address files easily!
- Supports the shift key modification if made to your Apple.
- Lets you work with your text on a screen at a time basis - reducing typos and allowing you to see your document as you edit it.
- Works with any printer!
- Supports the language card or any 16K expansion Ram card to keep more of your document readily available in memory.

SuperScribe is 100% machine language and requires only a 48K Apple II or II Plus with a disk drive. It may be purchased through your local computer store or direct from us by sending $89.95 plus $2.50 to cover shipping to:

ON-LINE SYSTEMS · 36575 Mudge Ranch Road · Coarsegold, CA 93614 · 209-683-6858

C.O.D., Master Charge or Visa accepted

Very neat. But this way you have no control over the direction of motion of the curtain, nor over the color to which the screen is cleared. Perhaps for your particular application neither of these restrictions makes any difference, in exchange for the very real advantages in speed and simplicity.

If you'd like to have your cake and eat it too, this can be arranged by POKEing a short machine-language subroutine into memory. Then you will be able to select your background color and still retain the speed advantage of the Monitor subroutine. You don't have to know anything about machine-language to do this, although for those who are curious I'll explain how it works in a few minutes. For the moment, just try the Applesoft program below:

```
10 GOSUB 1000 : REM PAINT SCREEN
20 FOR I = 768 TO 782 : REM POKE M/L SUB
30 READ J : POKE I,J
40 NEXT I
50 COLOR = C : REM YOUR CHOICE OF COLOR
60 CALL 768 : REM CALL THE SUBROUTINE
70 END
200 DATA 160, 39, 132, 45, 160, 39, 169, 0, 32, 40, 248,
   136, 16, 248, 96
```

For full-screen graphics, replace the second number in the DATA statment ("39") by the number "47".

If you RUN this program you'll see that it works just like the Monitor version, except that now the screen clears to the selected color, C, instead of only to black (C = 0). It should perhaps be pointed-out that once you have POKEd this subroutine into the computer by executing lines 20 through 40, you can CALL it any number of times in your program without having to POKE it in again. Lines 20 - 40 only have to appear and be executed once in each session at the computer.

Although quite fast, this screen-clearing operation is by no means instantaneous: you can still perceive a curtain-like movement across the screen. What if that's not good enough? I recently wrote a game program in which I wanted the screen to flash suddenly white, to indicate that an enemy torpedo had broken through my screens and wiped me out. Even the machine-language routines are too slow to make a believable explosion flash — an instantaneous white-out. Well, this can in fact be done with the help of a somewhat longer machine-language subroutine which I will now describe. And if you're not into writing game programs, you might still like to be able to clear your screen instantaneously to provide nice sharp transitions from one scene to the next.

The new program looks like this:

```
 70 FOR PAUSE = 0 TO 2000 : NEXT PAUSE
 80 GOSUB 1000 : REM REPAINT SCREEN
 90 FOR I = 800 TO 844 : REM NEW SUB
100 READ J : POKE I,J
110 NEXT I
120 COLOR = C
130 CALL 800 : REM CALL NEW SUBROUTINE
140 END
300 DATA 165, 48, 160, 120, 32, 45, 3, 160, 80, 32, 61,
   3, 96, 136, 153, 0, 4, 153, 128, 4, 153, 0, 5, 153, 128, 5
310 DATA 208, 241, 96, 136, 153, 0, 6, 153, 128, 6, 153, 0,
   7, 153, 128, 7, 208, 241, 96
```

For full-screen graphics, replace the ninth number in DATA statement 300 ("80") by the number "120"

As before, once this new subroutine has been POKEd into memory it can be CALLed whenever you need it without having to rePOKE it (unless, of course, you happen to overwrite it in the meanwhile). This subroutine has been deliberately placed into different memory locations than the previous one, so they can coexist in your computer. Furthermore, the Applesoft routines associated with these two different methods were written in such a way that when both have been typed into your computer as indicated, they will run consecutively. When you type RUN, the screen first fills up with colors, pauses for a few seconds, and then is erased by the first machine-language subroutine. Then the screen fills up with a new random color pattern, pauses, and is suddenly cleared by the second subroutine. The speed difference between these two subroutines is readily apparent in operation.

Each of the several different screen-clearing methods which have been described above has its own special properties; they are all useful additions to your programming arsenal.

Now, for those who are interested, let me briefly discuss the functioning of the two machine-language subroutines. I will assume that the reader is at least somewhat familiar with 6502 Assembly Language and its standard notation.

The first subroutine, starting at location 768 decimal (equivalent to $0300 in hexidecimal) is just a very slightly altered version of the Monitor's routine which we used earlier by CALLing -1994. The Monitor version clears the screen by drawing vertical black lines one after another, exactly as we did it in our very first Applesoft program. The difference in speed between these routines simply reflects the well-known speed advantage of machine-language over Basic. Since the Monitor's version only paints in one color — black — it was changed to permit the color to be an input variable using the standard Applesoft COLOR = C instruction to define which one you want. In Assembler notation, this subroutine looks like this:

```
$0300: A0 27    BKGRND LDY #$27       ; Maximum Y for mixed-screen
                                        clear
 0302: 84 2D           STY V2         ; Store as line-bottom coordinate
 0304: A0 27           LDY #$27       ; Rightmost X-coord (column)
 0306: A9 00    CLRSCR LDA #$00       ; Will start clearing at top
 0308: 20 28 F8        JSR VLINE      ; Jump to line-drawing subroutine
 030B: 88              DEY            ; Next leftmost X-coord (column)
 030C: 10 F8           BPL CLRSCR     ; Loop until done
 030E: 60              RTS            ; Done. Return
```

For full-screen graphics, the number "27" in location $0301 is replaced by the (hexidecimal) number "2F".

The alert reader may have noticed that the color to be used did not appear anywhere in this

PROGRESSIVE SOFTWARE

Presents

Software and Hardware for your APPLE

VU #1

- Accepts data from VISICALC*
- Relocates data and names
- Connects data and names
- Creates an out file that can be loaded into VISICALC*

Written by: MARC GOLDFARB $49.95
Add $4.95 for Postage and Handling.

VU #2

Includes all of the features of VU #1. VU #2 is a utility program for VISICALC* that has a built-in execute file that operates the program automatically and allows the user to do repetitious functions, such as payroll computing, etc. Written by:
MARC GOLDFARB $59.95
Add $4.95 for Postage and Handling.

VU #3

Includes the abilities of VU #1 and VU #2 but has additional superb features. This program will allow the user to enter data into VISICALC* from any program merely by inserting data into an array (which is well documented in the instructions). Then the program places the array into VISICALC*. VU #3 will also transfer data generated from VISICALC* into any of the user's programs via an array (as defined in the instructions). Written by:
MARC GOLDFARB $69.95
Add $4.95 for Postage and Handling.

Menu Planner — Has many types of dishes, counts calories for each entry. Prints out or displays menus with daily calorie count. Written by:
GARY A. FOOTE $19.95

Shopping List — Creates a list that can be used for food shopping or any other group of items. Allows user to create and save files of other items Prints out list. Written by:
CHARLES HEARN $19.95

Apple Menu Cookbook — Index-accessed data storage/retrieval program. Recipes stored, unlimited lines per entry. Easy editing. Formulated after NY Times Cookbook. Other useful features included. Written by:
WM. MERLINO, MD $19.95

Data Organizer — Alphabetize and maintain any data list up to 1,000 entries. Searching feature. Written by:
ED HOWERTON $14.95

*VISICALC is a trade-mark of Personal Software, Inc.

- Programs accepted for publication.
- Highest royalty paid.

FILES

- Builds Serial Files
- Changes Serial Files to Random Access Files
- Adds to End of Serial Files
- Record Insertion and Deletion Anywhere in Serial File
- Move Individual Records or Blocks of Records within Serial Files

A File manipulator that allows the user to search for a string within a file, sort data by blocks, handle many files at a time (without exiting the program) and saves executed files. A file can be saved under many names, viewed in several modes, and dumped (totally or partially) to a printer.

If you would like to work with files you must own this program! Comes with 10-page documentation in a binder. Because of the size and weight of this program, postage and handling charge is necessary. Written by:
MARC GOLDFARB FILES-Disk only. $49.95
Add $4.95 for Postage and Handling.

Super Sort and Search — A free-form data base package which allows the user to define fields per record. As the data is loaded in, it is automatically sorted. Can also merge together files or allow key sort entries. Will sort out in particular field, do a generic search and/or search for a specific piece of data. Will allow each particular record or records to be printed. Written by:
MARC GOLDFARB $69.95
Add $4.95 for Postage and Handling.

Matching Patterns — Choose two colored patterns that are exactly the same. Music response on correct patterns. Higher degree of accuracy produces longer melody before changing to more complex pattern (4 levels). Increases awareness of contrast and comparison. Written by:
JUDY PEGG Disk $19.95

TO ORDER

Send Check or Money Order to:

P. O. Box 273
Plymouth Meeting, PA 19462

Pennsylvania residents add 6% sales tax.

POSTAGE AND HANDLING

Please add $1.50 for the first item and $1.00 for each additional item.

U.S. and foreign dealer and distributor inquiries invited.

All programs require 16K memory unless specified.

215-279-4438

subroutine. In fact, the Applesoft statement COLOR = C automatically stores the appropriate color constant in location $30 (decimal 48), where the Monitor routine VLINE can get at it. VLINE draws a single vertical line of the specified color.

Now, the flash-clear subroutine beginning at location 800 decimal ($0320 hexidecimal) works by taking advantage of the "memory-mapped" nature of the Apple's low-resolution screen. Each of the 1600 screen positions on the mixed screen or the 1920 screen positions on the whole screen is defined by a specific half-byte (four bits, or one "nybble") in memory. Since these four bits can represent one of sixteen different hex numbers ($0 through F), each screen position will have one of sixteen different colors depending on how the defining nybble has been set. The two nybbles in each byte define the color for two screen positions in the same column but consecutive rows, that is, two vertically-stacked colored squares. To color a given square it is only necessary to find its corresponding nybble and set it to the appropriate value.

Unfortunately, for some reason the Apple designers didn't arrange the memory locations in any simple consecutive fashion to correspond to the screen rows in numerical order. It requires a special algorithm to find the byte which represents the first square of each row; all the rest of the squares in that row will be represented by consecutive bytes after that. To further complicate matters, the last eight bytes in every 128 bytes do not correspond to any screen positions at all, but rather are used as "scratchpad" memory for whatever devices might be in the motherboard slots.

This last little detail makes the required subroutine for clearing the screen much more complicated than it would otherwise have to be. It is necessary to take the byte in location $30, which represents the chosen color nybble repeated twice, and store it in each byte of screen memory, being careful not to disturb those special bytes which are possibly being used as scratchpad. The address of the first and last effective byte of each row in screen memory has to be known in advance in order to perform this operation in the fastest possible time, without taking time to compute these addresses during the operation. All this has been done in the algorithm represented by the assembly-language subroutine below:

```
$0320: A5 30    FLASH   LDA COLOR  ; Get selected color byte
 0322: A0 78            LDY #$78   ; Prepare to fill 120 bytes
 0324: 20 2D 03         JSR FILL1  ; Fill four sets of 120 bytes each
 0327: A0 50            LDY #$50   ; Prepare to fill 80 bytes
 0329: 20 3D 03         JSR FILL2  ; Fill four sets of 80 bytes each
 032C: 60               RTS        ; Done. Return.
      ; Subroutine FILL1 puts the selected color byte into
      ; each of four sets of 120 consecutive screen-memory
      ; bytes, being careful to avoid the scratchpad bytes at
      ; the end of each set.

 032D: 88       FILL1   DEY
 032E: 99 00 04         STA $400, Y
 0331: 99 80 05         STA $480, Y
 0337: 99 80 05         STA $500, Y
 033A: D0 F1            STA $580, Y
 033C: 60               BNE FILL1
                        RTS
      ; Subroutine FILL2 puts the selected color byte into each
      ; of four sets of 80 consecutive screen-memory bytes.
      ; These are the "short lines", leaving out at the end of
      ; each one of the four text lines at the bottom of the
      ; mixed screen.

 033D: 88       FILL2   DEY
 033E: 99 00 06         STA $600, Y
 0341: 99 80 06         STA $680, Y
 0344: 99 00 07         STA $700, Y
 0347: 99 80 07         STA $780, Y
 034A: D0 F1            BNE FILL2
 034C: 60               RTS
```

For full-screen graphics, the "short lines" of the subroutine FILL2 become full-length lines as in FILL1, which is accomplished simply by changing the constant "$50" in location $0328 to a "$78".

And that's how we clear the screen in a flash. But before I quit, I'd like to leave you with one more little idea. If, instead of setting the color byte by an Applesoft line of the form COLOR = C, you simply POKE into location 48 (decimal) any integer less than 256 (decimal), you may get a surprise. Depending on what integer you POKE, the screen may "clear" to a pattern of horizontal stripes! I'll bet that some clever reader out there will find some interesting and unexpected application for it. ©

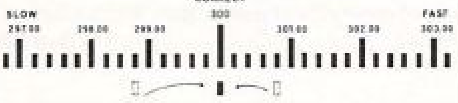
★★★★★★★★★★★★★★★★★★★

Bugs in your Apple?

DDT

Disco-Tech's Disc Drive Timer program

zaps disc drive problems!



Analyze disc drive motor speed on a routine basis with an adjustable real-time speedometer. Accurate to one-tenth of one r.p.m. out of 300 r.p.m.

Fine-tune disc drive motor speed yourself. All you need is DDT, two screwdrivers, and five minutes' time.

$29^{95} Postpaid
California residents add 6% tax
Diskette & complete manual

Engineering Business
Architecture Utilities
Surveying

Also available for TRS-80 Model I.

To order or for more information, write or call

disco tech T.M. microcomputer products a division of

Morton Technologies, Inc.
1150 Coddingtown Center
P.O. Box 11129 • Santa Rosa, CA 95406
707/523-1600
Dealer inquiries invited

★★★★★★★★★★★★★★★★★★★

Fun With Apple and Pascal

Gene A. Mauney
Greensboro, NC

While using Kenneth Bowles' excellent textbook, **Problem Solving Using PASCAL,** to self teach Pascal, it occurred to me to write this game program and make learning Pascal even more exciting. Since completing this writing I have discovered that Bowles' 1980 book, **Beginner's Guide for the USDA Pascal System,** would have helped and I am sure will be helpful with my next Pascal ventures.

I tried to use as many of the Apple-Pascal graphics functions as feasible in order to gain experience with these and of course depended on the **Apple Pascal Reference Manual** for this. From TURTLEGRAPHICS used are: MOVE, MOVETO, TURN, TURNTO, GRAFMODE, TEXTMODE, VIEWPORT, FILLSCREEN, TURTLEX, TURTLEY, WCHAR, and CHARTYPE. And from APPLESTUFF the RANDOM, PADDLE, BUTTON, and NOTE functions.

My plan was to use as much as would fit in with my study of the beginning lessons in Bowles' textbook along with developing a program for a game suggested to me by Peter Hildebrandt, to whom goes my appreciation. Also thanks to Bill Stanley for his helpfulness. In these beginnings I found that it would have been very helpful to have had some real Apple-Pascal programs for examples. So my hope is that this real program will be helpful for those readers who are beginners as I. No claims are made as to the most efficient methods for programming and I am sure that others will be able to find improvements. I will be happy to hear from anyone who has comments and suggestions. I hope programmers and players will enjoy it.

The Program

BEGIN(*MAIN*) first draws the Pentagon War Games frame using the TURTLE, then proceeds to the MOVEPENT PROCEDURE. The program switches back and forth between MOVEPENT and IFPADDLE. MOVEPENT creates the pentagons beginning at a random start point (AX,AY) with SIDE = 1, and moving from there in random ways increasing by SIDE + 3 (*NOTE6*) each time for nine times. Here is a place to change the difficulty level for the player if you wish. NINE counts the times through to know when nine pentagons have been formed and also to know the score for adding up totals. IFPADDLE accesses the paddle position and moves the gun. At two places (*NOTE 4*) the TURTLEGRAPHICS procedure, CHARTYPE(6), is used to turn off the previous position of the gun and bullets by XORing the image. CHR(11) is the up arrow used for gun and bullet. If BUTTON(0) is pushed so is TRUE, the IFBUTTON PROCEDURE produces the four bullets with sound each. Hit or miss is determined (*NOTE 5*) by using the last value of X, the lower left corner of the pentagon and the last value of SIDE along with the paddle position. If a hit is made, NINE, SCORE and TSCORE are added up, destruction of the pentagon is shown along with sound (*NOTE 3*), and the message shown. The TURTLEX and Y functions are used (*NOTE 2*) to determine the X,Y value of the pentagon corner for the destruction picture and 20 lines are used here. The procedure FILLSCREEN is used (*NOTE 1*) to erase the last pentagon just before the destruction image. Finally, after five pentagon attacks, the end message is shown along with the total score.

```
PROGRAM PENTAWAR;
USES TURTLEGRAPHICS, APPLESTUFF;
VAR   SCORE,TSCORE,X,Y,
      NINE,SIDE,PENTA: INTEGER;


PROCEDURE THEEND;
BEGIN
  TEXTMODE;WRITELN;WRITELN;
  WRITELN('* * *   PENTAGON WARS   * * *  ');
  WRITELN;WRITELN;
  WRITELN('     YOUR TOTAL SCORE IS ',TSCORE);
  WRITELN;WRITELN;WRITELN;
  WRITELN('DIRECTIONS: ');
  WRITELN('YOU WILL SEE 5 PENTAGON ATTACKS.');
  WRITELN('YOU WILL GET ONLY 5 SHOTS.');
  WRITELN('MAXSCORE IS 45 IF YOU HIT THE');
  WRITELN('SMALLEST PENTAGON OF EACH ATTACK.');
  WRITELN('(9,8,7,..0 AS PENTAGONS ATTACK.)');
  WRITELN('USE APPLE GAME PADDLE 0.  ');
  WRITELN;WRITELN;WRITELN;
  WRITELN('    PRESS RETURN THEN R FOR');
  WRITELN;
  WRITELN('    A NEW GAME.  GOOD LUCK!');
  WRITELN;
END;


PROCEDURE MISS;
VAR  TIME:  INTEGER;
BEGIN
  TEXTMODE;WRITELN;WRITELN;
  WRITELN; WRITELN;WRITELN;
  WRITELN('         YOU MISSED !');
  WRITELN;
  WRITELN('  ONLY ONE SHOT PER ATTACK.');
  WRITELN('   BETTER LUCK NEXT TIME.');
  WRITELN;
  WRITELN('  PRESS BUTTON TO CONTINUE.');
  WRITELN; WRITELN;WRITELN;
  WRITELN; WRITELN;WRITELN;
  FOR TIME:= 1 TO 800 DO
  BEGIN END;        (*WAIT BUTTON RELEASE*)
```

```
    REPEAT NINE:= 0
    UNTIL BUTTON(0);
  FOR TIME:= 1 TO 200 DO
  BEGIN END;                  (*WAIT AGAIN*)
END;

PROCEDURE HIT;
VAR  HITS,LENGTH,ANGLE,
     TX,TY,PITCH,DUR,TIME:   INTEGER;
BEGIN
  SCORE:= NINE + 1;
  TSCORE:= TSCORE + SCORE;
  VIEWPORT(2,277,90,180);
  FILLSCREEN(BLACK);          (*NOTE 1*)
  VIEWPORT(0,279,0,191);
  MOVETO(X,Y); TURNTO(90);
  LENGTH:=21; ANGLE:=120;
  DUR:=1; PITCH:=40;
  FOR HITS:= 1 TO 20 DO       (*NOTE 2*)
    BEGIN
      PENCOLOR(WHITE);
      MOVE(LENGTH);
      TX:=TURTLEX;  TY:=TURTLEY;
      PENCOLOR(BLACK);
      TURN(180); MOVE(LENGTH);
      MOVETO(TX,TY);
      TURN(ANGLE);
      LENGTH:=LENGTH-1; ANGLE:=ANGLE-2;
      NOTE(PITCH,DUR); PITCH:=PITCH-2;
    END;                      (*NOTE 3*)
    TEXTMODE;
    WRITELN;WRITELN;WRITELN;WRITELN;
    WRITELN('          A HIT ! ! !');
    WRITELN;WRITELN;
    WRITELN('          SCORE IS ',SCORE);
    WRITELN;WRITELN;
    WRITELN('   PRESS BUTTON TO CONTINUE.');
    WRITELN;WRITELN;
    WRITELN;WRITELN;WRITELN;WRITELN;
    FOR TIME:= 1 TO 800 DO
    BEGIN END;        (*WAIT BUTTON RELEASE*)
      REPEAT NINE:= 0
      UNTIL BUTTON(0);
    FOR TIME:= 1 TO 200 DO
    BEGIN END;                (*WAIT AGAIN*)
END;

PROCEDURE IFBUTTON;
VAR  PX,BUL,TWO,PITCH,DUR:   INTEGER;
BEGIN
  PX:= (PADDLE(0)+19);
  MOVETO(PX,20);
  FOR PITCH:= 40 TO 50 DO
  BEGIN
    DUR:=1; NOTE(PITCH,DUR);
  END;
  FOR BUL:= 1 TO 4 DO     (*4 BULLETS*)
    BEGIN
      TURNTO(90);             (*TURN UP*)
      MOVE(20);
      FOR TWO:= 1 TO 2 DO
        BEGIN
          CHARTYPE(6);        (*NOTE 4*)
          WCHAR(CHR(11));
          TURNTO(180);
          MOVE(7);
        END;
    END;
    IF (PX > X) AND (PX < (X+SIDE))
      THEN BEGIN HIT;
      END                     (*NOTE 5*)
      ELSE BEGIN MISS;
      END;
END;
```

Apple Monitor Extender

APPLE II 16K, CASSETTE

This utility program works in complete harmony with the Apple monitor to extend your computer's capability and help you use the full power of machine language programming.

Screen display shows memory in HEX, ASCII or BINARY. Move data anywhere in memory without regard to direction or overlapping and read or write any sector on disk. Insertions may be in HEX or ASCII so you can easily format high speed text displays without conversions.

Study, modify or dis-assemble any program, complete with labels. Several programs may be combined, and the entire disassembled text file stored on disk/tape for later assembly.

The slow listing feature steps through listings with ease.

© Copyright 1980 Glenn R. Sogge, All Rights Reserved.

IMAGE COMPUTER PRODUCTS™ 615 Academy Drive Northbrook, IL 60062 312/564-5060

Computer House Division

PROGRAMS FOR COMMODORE AND APPLE

Legal accounting Demo	$15.00
Legal accounting Program	995.00
Machine Part Quote Demo	15.00
Machine Part Quote Program	325.00
Mailing/phone list	80.00
Political Mail/phone list	130.00
Beams, structural	115.00
Trig/Circle Tangent	110.00
Spur Gears	35.00
Bolt Circles	25.00
Filament Wound TAnks	125.00
Scrunch	25.00

PROGRAMS FOR COMMODORE ONLY

A/P, A/R, Job Cost & Job Est.	370.00
Inventory	95.00
Financial	175.00
Real Estate Listings	265.00
Check Writer	25.00
File Editing Tools (FET)	65.00
Screen Dump/Repeat	35.00
Docu-Print	20.00
Scrunch	25.00
Sof-Bkup	40.00
Sorter (Mach. Language)	35.00
Trace-Print	25.00
Vari-Print	25.00

ASK FOR CATALOG #80-C2 Dealers Wanted
Computer House Div. 1407 Clinton Road
Jackson, Michigan 49202 (517) 782-2132

```
PROCEDURE IFPADDLE;
VAR  TIME:  INTEGER;
BEGIN
  PENCOLOR(BLACK);
  MOVETO((PADDLE(0)+19),20);
  FOR TIME:= 1 TO 8 DO          (*TIME*)
  BEGIN                        (*ADJUST*)
    CHARTYPE(6);
    WCHAR(CHR(11));            (*NOTE 4*)
    TURNTO(180);
    MOVE(7);
  END;
  IF BUTTON(0)  THEN
  BEGIN
    IFBUTTON;
  END;
END;



PROCEDURE MOVEPENT;
VAR  DRAW,EACHONE,
          AX,BX,CX,NX,AY,BY:  INTEGER;
BEGIN
  REPEAT
  PENTA:= PENTA+1; SCORE:= 0;
  NINE:= 9; SIDE:= 1; BY:= 1; NX:= 1;
  AX:= 40+RANDOM MOD(200);
  AY:= 166;
  PENCOLOR(BLACK);
  MOVETO(AX,AY);         (* PENTAGON START*)
  WHILE NINE > 0 DO     (*9 PENTAGONS EACH*)
  BEGIN
    VIEWPORT(1,278,90,180);
    FILLSCREEN(BLACK);      (*CLEAR SCREEN*)
    VIEWPORT(0,279,0,180);
    GRAFMODE;
    BX:= RANDOM MOD(6+NX);
    CX:= RANDOM MOD(6+NX);
    BX:=BX-CX; BY:=BY+8; NX:=NX+4;
    X:=AX-BX; Y:=AY-BY;
    SIDE:= SIDE+3;              (*NOTE 6*)
    NINE:= NINE-1;
    PENCOLOR(BLACK);
    MOVETO(X,Y); TURNTO(0);
    PENCOLOR(WHITE);
    FOR EACHONE:= 1 TO 5 DO
    BEGIN                       (*5 SIDES*)
      MOVE(SIDE);
      TURN(72);         (*PENTAGON ANGLE*)
    END;
    IFPADDLE;
  END;
  UNTIL PENTA = 5;     (*AT END OF GAME*)
  THEEND;
END;



BEGIN                    (*MAIN*)
  INITTURTLE;
  MOVETO(0,0);
  PENCOLOR(WHITE);   (*DRAW THE FRAME*)
  MOVETO(279,0);
  MOVETO(279,191);
  MOVETO(0,191);
  MOVETO(0,0);
  PENCOLOR(BLACK);
  RANDOMIZE;
  PENTA:= 0; TSCORE:= 0;
  MOVEPENT;
  READLN;                (*WAIT FOR <RTN>*)
END.
```

©

OUR NEW PROGRAM PACKAGE
FOR THE APPLE II
LOOKS TERRIFIC

SO DOES THE PRICE

Can 25 quality Apple programs cost a mere $24.95?

We have a lot to gain by almost losing our shirts. Because we are a new and growing company, we need your business to build our business.

APPLE SACK I — Twenty-five Applesoft programs with everything from hi-res graphics to adventures. Programs include 5 business, 5 utility, 10 games and 5 home programs.

APPLE SACK II — TV TRIVIA — Hundreds of questions about you r old favorite TV shows from the 50's and the 60's. Extra points given for super stumpers. Will tingle your memory cells. (Integer Basic)

APPLE SACK III — ADVENTURES — 8 Adventures for excitement in your life. Travel to far away places, thru space, into caves, explore dungeons and fight for your life.

EACH PACK IS ONLY $24.95 AND INCLUDES SHIPPING — GET ALL THREE PACKAGES FOR $49.95.

It's not that our program packages for the Apple are so cheap. It's just that the other companies are so expensive.

ORDER YOUR SACK TODAY FROM . . .

COMPUTERMAT
Box 1664C
Lake Havasu City, Arizona 86403
(602) 855-3357

NEW FOR APPLE 2 & APPLE 2 Plus

PLANETOIDS

A real time high resolution, graphics game with super sound effects. Features three different skill levels - Novice-for beginners, Average-"Arcade" difficulty and Expert-where the asteroids are attracted to your ship. Has true momentum movements & special design your own asteroid belt option! (NOTE: previous "Asteroids" owners - send in your original tape or disk [with $3 postage and handling] for upgrade to Planetoids.

PLANETOIDS
Formerly "ASTEROID" a T.M. of Atari

Apple 2 - 32K tape
041-0015 $14.95

Apple 2 - 32K disk
042-0015 $19.95

NOW AVAILABLE FROM YOUR LOCAL COMPUTER STORE OR SEND TO A.I.
SEND FOR YOUR FREE CATALOG

© Copyright 1980
Adventure International
A DIVISION OF SCOTT ADAMS, INC

BOX 3435
LONGWOOD FLA 32750
(305) 862-6917

DEALER INQUIRIES INVITED

Flipping Your Disk

M. G. Sieg

If you own an APPLE DISK II, you can double the storage capacity of a single mini-floppy at virtually no cost. The only things you need are at least two floppies, a hand held hole punch, and a colored pencil that will show on black.

The trick is simple, make your single sided floppy into a dual sided ''flipped'' floppy.

First, let's get acquainted with the anatomy of a floppy disk. Externally there is a black jacket with several holes cut into it. The inside of the jacket is lined with a white fabric which can only be seen by prying the jacket apart a bit at the center hole. Through the holes, the rust colored disk can be seen. The rust color is a coating on a mylar surface enabling the drive to read and record information much the same as the tape for your cassette recorder.

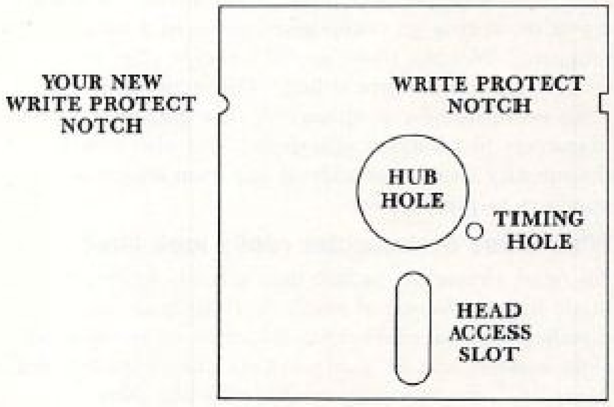
The hole in the center of the jacket is the hub hole. This permits the disk drive motor to engage the disk and spin it. The long wide slot just below the hub hole is called the head access slot. It permits the read/write head and the pressure pad to access the spinning disk. IMPORTANT: Avoid touching the disk surface through this slot. Fingerprints on the recording surface in this area can cause I/O errors. Just to the right of the hub hole is a small hole through which the disk surface can be seen at times or, at other times, a hole completely through the other side of the jacket appears. This is the timing hole. Finally, in the upper right corner (if you consider the head access slot the bottom) of the jacket, there is a rectangular slot. This is the write protect notch. When the floppy is inserted into the drive, a mechanical switch can slip into this notch signalling the drive that it is OK to write on this disk. If the notch is not present or is covered with a piece of tape the disk is ''write protected'' thereby preventing the APPLE from writing anything on this disk — even the initialization information.

By duplicating this write protect notch at the same position on the left side of the jacket, the disk can be turned over and the DISK II may write on the ''flip'' side. The APPLE DISK is different than most other drives because it ignores the timing hole, using the motor and 'soft' timing techniques instead.

If you follow these instructions carefully, a good 90% of major brand mini-floppies can be turned into ''flipped'' floppies. Place two disks in front of you face up on a very clean surface. Once again you are cautioned not to touch the recording surface through the head access slot. Take one of the disks and place it flipped over on top of the other, such that the head access slots are at the bottom. Align them both exactly and, with a light colored pencil, make a mark on the bottom disk along the inside edge of the flipped floppy's write protect notch. With a standard hole puncher, punch a half hole (i.e. no further into the jacket than your pencil mark) completely through both sides of the jacket at your pencil mark. This half hole is now the write protect notch for the flip side. The fact that this hole is round is of no consequence, since the only thing of importance is that the mechanical switch inside your drive can drop into a notch of some type.

Test your ''flipped'' floppy by inserting it into the drive (flipped side up naturally) and doing the normal INIT procedure. If you get several groans from your drive followed by an I/O ERROR, you may not have your notch deep enough or you may have run into one of the 10% or so disks that have flaws in the flipped surface. If you suspect your notch may not be deep enough, very carefully cut away a little more of the jacket. You must be careful not to cut into the disk surface, for that may ruin the disk completely. Assuming you have reasonable quality disks, the flipped surface having a flaw will be a rare problem but has no solution. If you should be unlucky enough to have this occur first time out, don't be discouraged; try another disk.

These flipped floppies may now be used exactly as you use all the normal disks in your collection.



This is what your flipped floppy should look like *after* following the procedures.

Editor's Note: While we've printed this article as a reader service, you should be ***well aware*** *of the risks involved. Disks made for single sided use may contain flaws on the reverse side. We can't vouch for the author's 10% figure. In essence, try this at your risk! RCL*

©

Turn your Apple into the world's most versatile personal computer.

The SoftCard™ Solution. SoftCard turns your Apple into two computers. A Z-80 and a 6502. By adding a Z-80 microprocessor and CP/M to your Apple, SoftCard turns your Apple into a CP/M based machine. That means you can access the single largest body of microcomputer software in existence. Two computers in one. And, the advantages of both.

Plug and go. The SoftCard system starts with a Z-80 based circuit card. Just plug it into any slot (except 0) of your Apple. No modifications required. SoftCard supports most of your Apple peripherals, and, in 6502-mode, your Apple is still your Apple.

CP/M for your Apple. You get CP/M on disk with the SoftCard package. It's a powerful and simple-to-use operating system. It supports more software than any other microcomputer operating system. And that's the key to the versatility of the SoftCard/Apple.

BASIC included. A powerful tool, BASIC-80 is included in the SoftCard package. Running under CP/M, ANSI Standard BASIC-80 is the most powerful microcomputer BASIC available. It includes extensive disk I/O statements, error trapping, integer variables, 16-digit precision, extensive EDIT commands and string functions, high and low-res Apple graphics, PRINT USING, CHAIN and COMMON, plus many additional commands. And, it's a BASIC you can compile with Microsoft's BASIC Compiler.

More languages. With SoftCard and CP/M, you can add Microsoft's ANSI Standard COBOL, and FORTRAN, or Basic Compiler and Assembly Language Development System. All, more powerful tools for your Apple.

Seeing is believing. See the SoftCard in operation at your Microsoft or Apple dealer. We think you'll agree that the SoftCard turns your Apple into the world's most versatile personal computer.

Complete information? It's at your dealer's now. Or, we'll send it to you and include a dealer list. Write us. Call us. Or, circle the reader service card number below.

SoftCard is a trademark of Microsoft. Apple II and Apple II Plus are registered trademarks of Apple Computer. Z-80 is a registered trademark of Zilog, Inc. CP/M is a registered trademark of Digital Research, Inc.

MICROSOFT
CONSUMER PRODUCTS

Microsoft Consumer Products, 400 108th Ave. N.E., Bellevue, WA 98004. (206) 454-1315

CIRCLE READER SERVICE NUMBER

Resolving Applesoft and Hires Graphics Memory Conflicts

Jeff Schmoyer

This article will attempt to divulge solutions to memory usage conflicts that can occur when an Applesoft program becomes large enough to start taking up residence in a Hires screen page. Of course the problems only appear when a program utilizing Hires graphics is executed. Throughout this article, numbers will be used in both decimal and hexadecimal (base 16). Hexadecimal numbers are represented with a dollar sign ($) preceding them, i.e. $800.

First, it is necessary to understand the memory layout of the region of RAM with which we are concerned. Applesoft programs may reside anywhere in memory from $800 to $BFFF in a 48K Apple II. If a disk is being used, the top boundary will be lower, generally $9600. The top boundary is not really important to this duscussion so it will be referred to as the top of memory. It makes no difference where it is.

The two Hires screens are in fixed positions in memory, the first located from $2000 to $3FFF, and the second from $4000 to $5FFF. Figure 1 is a map demonstrating what is known so far.

As may be seen in the drawing, if an Applesoft program is confined to the area from $800 to $1FFF, there is no conflict. This allows 6K of program space.

Now it's time to introduce another variable, variables! Not only does the program itself take up space, but as variables are allocated in the program, they too have to exist in memory.

String variables are no problem. They allocate space from the top of memory down. Plenty of unused space is available in this region.

Simple variables and numeric and string arrays on the other hand, start at the end of the program

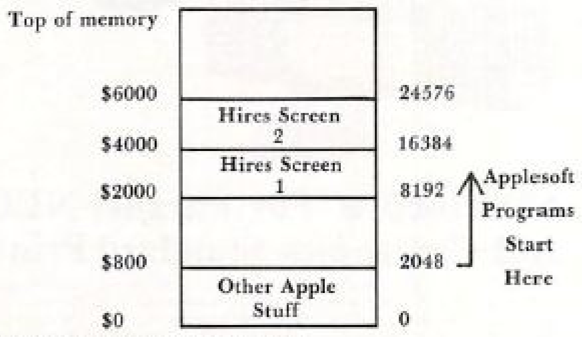


Figure 1: Partial Memory Map

and move up through memory. If there are enough of them, they will cruise right into the Hires page. If this happens, whenever the Hires screen is altered by an HGR or some other command, the variables in the screen space will be changed or erased!

At this point it can be seen that there is 6K of memory available for the program and simple variables and arrays altogether. Now the space is starting to get tight.

The first solution that comes to mind is simply to switch screens and use the second graphics page instead of Hires page 1. That will free up an additional 8K of memory yielding 4K total.

This is not really a bad way to go except that some Hires features are not fully supported for the second screen, such as the mixed text and graphics mode. For Hires page 2, the four lines of text at the bottom are always filled with garbage. (The lines are not actually full of garbage but that will have to be considered in a future article.) There is also the possibility of needing both Hires screens in the program for animation or some other purpose. So this solution may not be totally acceptable.

One acceptable solution deals solely with the variables. If it can be determined that the program itself does not infringe on the Hires territory, the variables may be dealt with separately. This determination may be made by loading the questionable program, entering HGR, entering TEXT, and then listing the program. (Make sure the program has been saved somewhere first.) If the end of the program is still intact, the program fits in the room available. If it is gone then move directly to the next solution. This one will not do it.

If the program fits and the variables do not, the variables may be easily moved to another region of memory. To do this, as the first line of the Applesoft program enter 0 LOMEN:16384. This line must be executed before any variables are allocated. In this way variables are stored above Hires page 1 and out of the way. If both screens usage are required, 0 LOMEN:24576 may be alternately entered to start variable allocation above screen 2. Be sure to check that the additional program line did not extend the program past the Hires boundary.

If the program itself is too large for the available space, it must be moved to a more roomy area of memory, in this case above the Hires pages. There are two page zero locations which control where an Applesoft program starts, $67 and $68. By altering these locations and reloading the program, it can be run from the new location. These alterations may be made from the direct mode or by a startup program. As long as Applesoft is not reinitialized the changes will remain in effect.

In reality only location $68 in page zero need be changed since $67 will be set to 1 in any case. The necessary poke is POKE 104,96. 104 translates to location $68, while the 96 in hex is $60. This is the high order byte of the new program starting address, $6001. Alternatively POKE 104,64 may be used to locate the program at $4001. This operation has set up the new address for the program to be loaded and run from. For the programs to execute correctly at the new location, one other poke is necessary. The location preceding the program must be set to zero. This would be $6000 for the program at $6001 or $4000 for the program at $4001. POKE 24576,0 or POKE 16384,0 respectively will accommodate this change for programs at $6001 and $4001.

After the program is moved, a 'dead zone' is left in memory from $800 to $1FFF. Neither the program nor any of its variables will use this space. Its best use would be for machine language routines and tables.

To reiterate, for a program to load and run above Hires page 1, POKE 104,64: POKE 16384,0 is necessary. For a program to load and run above Hires page 2, POKE 104,96: POKE 24576,0 is necessary.

Remember, the program will not actually be moved by this operation. Only programs loaded and run after this point will be above the Hires screen. Also, reinitialization of Applesoft will reset the pointers to $800. Setting LOMEM as with the previous technique is not necessary and should not be done.

To recap, three techniques to avoid memory conflicts with Applesoft and the Hires screens were outlined. The first is to use Hires page 2 instead of Hires page 1. The second is to move the simple variables and arrays out of the way with the LOMEM command. The third is to change the program start pointers to reset the program load and run point above the Hires pages. There are other ways to accomodate the screens but these few should suffice in most cases. ©

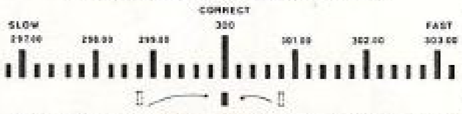
★★★★★★★★★★★★★★★★★★★

Bugs in your Apple?

DDT

Disco-Tech's Disc Drive Timer program

zaps disc drive problems!



Analyze disc drive motor speed on a routine basis with an adjustable real-time speedometer. Accurate to one-tenth of one r.p.m. out of 300 r.p.m.

Fine-tune disc drive motor speed yourself. All you need is DDT, two screwdrivers, and five minutes' time.

$29.95 Postpaid
California residents add 6% tax.
Diskette & complete manual.

Engineering Business
Architecture Utilities
Surveying

Also available for TRS-80 Model I.

To order or for more information, write or call

disco tech T.M.

microcomputer products
a division of
Morton Technologies, Inc.
1150 Coddingtown Center
P.O. Box 11129 • Santa Rosa, CA 95406
707/523-1600
Dealer inquiries invited.

★★★★★★★★★★★★★★★★★★★

LOOK!!!

AVANT-GARDE CREATIONS has SOFTWARE:

EDUCATION
ART/DESIGN
GAMES
BUSINESS
UTILITIES
SELF TRANSFORMATION

LOOK AGAIN!!!

We have the following and MORE!:

5 Great Games! Animal Bingo, Jungle Safari, Space Defense, Sky Watcher, Air Traffic Controller $29.95 (or $9.95 each)

5 More Great Games! Deep Sea Treasure, Mystery Code, Depth Charge, The Mine Fields of Normalcy, Turn 'Em Loose $29.95 (or $9.95 each)

The Mailing Label & Filing System Filing, label-making, binary sort, dynamic sorting, directory, quick-find, more! $24.95

Sentence Diagramming Educational, grades 6-12 $19.95

Action Sounds & Hi-Res Scrolling Designed to give your program the excitement of **action & sound** $15.95

Super Draw & Write Fonts, drawing, and useful utilities $15.95

Super Shape Draw! The best system yet, it works! ...creates shape tables like a dream... $19.95

The Creativity Package Draw, write poetry, music $19.95

"*...impressive...satisfying...interesting...fun!*" Peelings (The Magazine of Software Reviews)

"*Truly different...unique...the program is an enjoyable one...cute...very interesting... new...nice...a good value!*" Apple Orchard (Winter)

Demo Disk I Some of our best stuff $9.95

Demo Disk II More of our best $9.95

All of our software is written in Applesoft*, 48K, disk

AVANT-GARDE CREATIONS
P.O. Box 30160
Eugene, OR 97403 Dept. CM
(503) 345-3043
(12pm-6pm 7 days a week)

DEALER INQUIRIES INVITED
VISA/MASTERCARD

*Apple is a trademark of Apple Computer, Inc.

Fill The Screen With Your Message: Generating Large Multi-Colored Characters Using Apple Low-Resolution Graphics

Francis A. Harvey
Rosann W. Collins
Theodore C. Hines
School of Education
University of North Carolina at Greensboro
Greensboro, North Carolina 27412

Programs written by beginning programmers can often be distinguished from more elaborate "commercial" programs by the fact that the commercial programs make such extensive use of color and graphics. Computers such as the Apple and Atari have very good graphics capability, but many users lack the time or programming background, or both, to take full advantage of these capabilities. As a result their programs, while they may be carefully designed and interesting, lack the pizzazz that children expect from computers as a result of their experiences with commercial programs and computer games at home and in game rooms.

As part of a series of utilities of this kind, we have developed a set of subroutines in Applesoft which will display the characters in any string on the screen as large, colorful letters. With these subroutines program instructions, prompts, positive reinforcement, and negative responses to user input can look just like those in "real" computer games. Very little modification of an existing program is required to convert screen output to this form.

Each character is defined (with a combination of PLOT, VLIN, and HLIN commands) on a matrix which uses seven blocks in the vertical dimension and which varies in width depending on the shape of the character. With the character set defined in this way, each line of text can have between six and nine characters, and a total of four lines of text can be displayed. Each letter is approximately one-fifth as

```
10  REM   ---- LETTER  MATCH  ---
20  REM   ---- 10/20/80 VERSION--
30  REM --------  BY   --------
40  REM ---- FRANCIS A. HARVEY -
50  REM ---- ROSANN H. COLLINS -
60  REM -- & THEODORE C. HINES -
65  REM --- COPYRIGHT 1980 -----
70  HOME : GR : GOSUB 5020
80  REM -------TITLE PAGE------
90 Y = 3:A$ = "MATCH": GOSUB 6010
100 Y = Y + 12:A$ = "THE": GOSUB 6010
110 Y = Y + 12:A$ = "LETTERS": GOSUB 6010
120  PRINT : PRINT : PRINT
130  FOR I = 1 TO 1000: NEXT I
141  PRINT "BY        FRANCIS A. HARVEY"
142  PRINT "          ROSANN H. COLLINS"
143  PRINT "          THEODORE C. HINES"
145  FOR I = 1 TO 2500: NEXT I
146  PRINT : PRINT : PRINT
160 A$ = "COPYRIGHT OCTOBER 1980": GOSUB 4020
170  FOR I = 1 TO 4000: NEXT
180  PRINT : PRINT : PRINT : REM --CLEAR TEXT
190 E = 5: REM --- FOR DEMO PURPOSES
200  REM :E IS NUMBER OF ATTEMPTS
210  REM --- USER INSTRUCTIONS---
220  GR : GOSUB 5020
230 X = 0:Y = 0: REM --RESETS LETTER POSITION
240 A$ = "I TYPE": GOSUB 6010
250 Y = Y + 12
260 A$ = "A": GOSUB 6010
270 Y = Y + 12:A$ = "LETTER.": GOSUB 6010
280  FOR I = 1 TO 5000: NEXT I: GR
290  GOSUB 5020
300 Y = 3
310 A$ = "YOU TYPE": GOSUB 6010
320 Y = Y + 9
330 A$ = "THE SAME"
340  GOSUB 6010
345 Y = Y + 9:A$ = "LETTER.": GOSUB 6000
350  FOR I = 1 TO 4000: NEXT I
360 Y = Y + 10
370 A$ = "READY?": GOSUB 6000
380  HOME
390  INPUT "STRIKE 'RETURN' WHEN READY.";A$
400  REM
410  REM *********************
420  REM ---BEGIN MAIN PROGRAM---
430 L =  RND (1) * 26 + 1
440 C1 = C1 + 1: REM --COUNTS LETTERS TRIED
450 L =  INT (L) + 64
460 Y = 3
470 A$ =  CHR$ (L): GR : GOSUB 5020: GOSUB 6010
480  HOME
490  PRINT "TYPE THE SAME LETTER."
500  GET B$
510  REM ---DISABLE RETURN KEY
520  IF  ASC (B$) = 13 THEN 500
530  REM ---DISABLE SPACE BAR
540  IF  ASC (B$) = 32 THEN 500
550 Y = Y + 8
560 A$ = B$: GOSUB 6010
570  FOR K = 1 TO 500: NEXT
580  IF B$ =  CHR$ (L) THEN  GOSUB 2010
590  IF B$ <  >  CHR$ (L) THEN  GOSUB 1010:Y = 3: GOTO 500
600  FOR I = 1 TO 2000: NEXT I
610  IF C1 < E THEN 420
620  GR : GOSUB 5000:Y = 3
630  HOME
640 A$ = "THAT'S": GOSUB 6010
650 Y = Y + 12
660 A$ = "ALL": GOSUB 6010
670 Y = Y + 12
680 A$ = "FOR NOW.": GOSUB 6010
```

high as the screen and about one-eighth of a screen wide.

Color can be set within the program or randomly selected each time a line of characters is displayed. The upper-left corner of the matrix is defined as (X,Y), and each character is "drawn" from this reference point.

Each line of characters is passed to the subroutine as the string A$. An initial value of X, the horizontal beginning point of each character, is calculated which will center the characters on the line, and the characters are drawn one at a time.

The subroutines that draw each character automatically increment the value of X the appropriate number of spaces to the right. Messages longer than one line (e.g., "You are sharp!") can be subdivided; the value of the string A$ is set to the contents of each line, and Y is incremented by at least nine before calling the subroutine which centers and plots the characters.

The sample program listed demonstrates two ways in which these techniques can be used. The program, LETTERMATCH, was developed to familiarize primary school students with the letters of the alphabet and the computer keyboard. A randomly selected letter is displayed on the screen, and the user is asked to type the same letter. The GET command is used for input and all non-letter keys, the RETURN key, and space bar are disabled.

If the student enters the wrong letter, the computer responds with a large "TRY IT AGAIN," then clears the screen of the student's response and redisplays the original letter. The student responds until the correct letter is selected. When the student does enter the correct letter, the computer responds (again, in large, multi-colored letters) with one of five randomly selected positive responses, such as "RIGHT!" or "YOU ARE SHARP!". Each student is asked to identify five letters correctly.

```
690  FOR I = 1 TO 2000: NEXT I
700  GR : GOSUB 5000: REM —CLEARS SCREEN
710 Y = 3
720 A$ = "NEXT": GOSUB 6000
730 Y = Y + 12:A$ = "PERSON,": GOSUB 6000
740 Y = Y + 12:A$ = "PLEASE.": GOSUB 6000
745  FOR I = 1 TO 2000: NEXT I
750  PRINT
760  PRINT "TYPE ";: FLASH : PRINT "S";: NORMAL : PRINT " TO STOP."
770  PRINT "STRIKE ANY KEY TO GO ON."
780  GET Z$: IF Z$ <  > "S" THEN C1 = 0: GR : GOTO 380
790  GR : GOSUB 5000:Y = 3
800  HOME
810 A$ = "OK!": GOSUB 6000
820 Y = Y + 12:A$ = "GOODBYE"
830  GOSUB 6000
840 Y = Y + 12:A$ = "FOR NOW.": GOSUB 6000
899  END
1000  REM
1010  REM ——SUBROUTINE FOR WRONG ANSWERS
1020 Y = Y + 8
1030 A$ = "TRY IT": GOSUB 6010
1040 Y = Y + 8:A$ = "AGAIN.": GOSUB 6010
1050  FOR I = 1 TO 1500: NEXT I
1060 Y = 11: GOSUB 3000
1070  RETURN
2000  REM
2010  REM ———SUBROUTINE FOR RIGHT ANSWERS
2020 Y = Y + 12
2030 H =  INT ( RND (1) * 5) + 1
2040  ON H GOTO 2050,2060,2070,2090,2100
2050 A$ = "RIGHT!": GOSUB 6000: RETURN
2060 A$ = "OK!2:GOSUB 6010: RETURN
2070 A$ = "YOU ARE": GOSUB 6010
2080 Y = Y + 8:A$ = "SHARP!": GOSUB 6010: RETURN
2090 A$ = "GREAT!2:GOSUB 6010: RETURN
2100 A$ = "SUPER!": GOSUB 6010: RETURN
2110  RETURN
3000  REM —BLANKS REST OF SCREEN
3010  COLOR= 0
3020  FOR T = Y TO 39
3030  HLIN 0,39 AT T
3040  NEXT
3050  GOSUB 5020
3060  RETURN
4000  REM
4010  REM *********************
4020  REM ——CENTERS AND PRINTS
4030  REM ——REGULAR TEXT——
4040 Z = (40 -  LEN (A$)) / 2
4050  HTAB Z: PRINT A$
4060  RETURN
5000  REM
5010  REM *********************
5020  REM —PICKS RANDOM COLOR——
5030  COLOR=  INT ( RND (1) * 15) + 1
5040  RETURN
6000  REM  **********************
6010  REM -LARGE PRINT SUBROUTINE
6020  REM ——A$ IS STRING TO———
6030  REM  ——  BE PRINTED ————
6040  REM ————CENTERS TEXT—
6050 X =  ABS (20 -  LEN (A$) * 2.5)
6060  FOR H = 1 TO  LEN (A$)
6070  IF  ASC ( MID$ (A$,H,1)) = 32 THEN X = X + 2: GOTO 6160
6080  IF  MID$ (A$,H,1) = "?" THEN  GOSUB 8010: GOTO 6160
6090  IF  MID$ (A$,H,1) = "!" THEN  GOSUB 8080: GOTO 6160
6100  IF  MID$ (A$,H,1) = "," THEN  GOSUB 8130: GOTO 6150
6110  IF  MID$ (A$,H,1) = "." THEN  GOSUB 8180: GOTO 6160
6120  IF  MID$ (A$,H,1) = "'" THEN  GOSUB 8230: GOTO 6160
6125  IF  MID$ (A$,H,1) = ";" THEN  GOSUB 8270: GOTO 6160
6130 P =  ASC ( MID$ (A$,H,1)) - 64
6140  ON P GOSUB 6200,6270,6350,6420,6490,6560,6620,6700,6750,6810,6860,69
    00,6950,7010,7070,7130,7190,7270,7360,7440,7480,7530,7590,7650,7700,7
    750
6150 X = X + 6
6160  NEXT
6170  GOSUB 5020
6180  RETURN
6200  REM ————————PRINTS LETTER A
6210  PLOT X + 2,Y
6220  PLOT X + 1,Y + 1: PLOT X + 3,Y + 1
```

We have found that LETTERMATCH provides an excellent introduction to computers for quite young children. It can be modified (for example, by considerably shortening the delay loops) for use to introduce other users to computers and to sharpen keyboarding skills of users of any age.

LETTERMATCH occupies about 6400 bytes of memory. The subroutines which format and plot the characters (beginning in line 5000) occupy about 3700 bytes. Thus, any Applesoft program which leaves 3700 bytes of memory free when loaded can be modified to give large, multi-colored responses by merging these subroutines with the existing program (using the Apple RENUMBER program), making minor changes in the main program (adding GR, selecting colors, etc.), then modifying each PRINT statement to use the subroutine. For example, the line "200 PRINT "VERY GOOD!" would be changed to "200 A$ = "VERY GOOD!": GOSUB 6000."

The character set as developed includes the upper case letters A to Z and the question mark, exclamation point, comma, period, single quotation marks, and semicolon. The set could easily be expanded to include lower case letters, numerals, and other punctuation. The program randomly selects the color of each line of characters.

Copies of LETTERMATCH on diskette or cassette are available from the authors at the above address for the cost of duplication. While we reserve all commercial rights to these programs, we offer them free to any user for any non-commercial educational purpose. Other utilities of this kind which we have developed include routines for adding sound effects and music to programs, additional graphics (such as screen borders), and others. These will appear in later issues of **COMPUTE!** We hope that teachers and other computer users will find these procedures a useful addition to their program collection.

```
6230  VLIN Y + 2,Y + 6 AT X + 4
6240  HLIN X,X + 4 AT Y + 4
6250  VLIN Y + 2,Y + 6 AT X
6260  RETURN
6270  REM ———PRINTS B
6280  VLIN Y,Y + 6 AT X
6290  HLIN X,X + 2 AT Y
6300  HLIN X,X + 2 AT Y + 3
6310  HLIN X,X + 2 AT Y + 6
6320  VLIN Y + 1,Y + 2 AT X + 3
6330  VLIN Y + 4,Y + 5 AT X + 3
6340  IF X > 0 THEN X = X - 1: RETURN
6350  REM ————PRINTS C
6360  VLIN Y,Y + 6 AT X
6370  HLIN X,X + 3 AT Y + 6
6380  HLIN X,X + 3 AT Y
6390  PLOT X + 3,Y + 1
6400  PLOT X + 3,Y + 5
6410 X = X - 1: RETURN
6420  REM ————PRINTS D
6430  VLIN Y,Y + 6 AT X
6440  VLIN Y + 1,Y + 5 AT X + 3
6450  HLIN X,X + 2 AT Y
6460  HLIN X,X + 2 AT Y + 6
6470 X = X - 1: RETURN
6480  PRINT "1799": END
6490  REM ——————PRINTS E
6500  VLIN Y,Y + 6 AT X
6510  HLIN X,X + 3 AT Y
6520  HLIN X,X + 2 AT Y + 3
6530  HLIN X,X + 3 AT Y + 6
6540 X = X - 1
6550  RETURN
6560  REM —————PRINTS F
6570  VLIN Y,Y + 6 AT X
6580  HLIN X,X + 3 AT Y
6590  HLIN X,X + 2 AT Y + 3
6600 X = X - 1
6610  RETURN
6620  REM ————————PRINTS G
6630  VLIN Y,Y + 6 AT X
6640  HLIN X,X + 3 AT Y + 6
6650  HLIN X,X + 3 AT Y
6660  PLOT X + 3,Y + 5: PLOT X + 3,Y + 4
6670  PLOT X + 2,Y + 4
6680 X = X - 1
6690  RETURN
6700  REM ————————PRINTS H
6710  VLIN Y,Y + 6 AT X
6720  VLIN Y,Y + 6 AT X + 3
6730  HLIN X,X + 3 AT Y + 3
6740 X = X - 1: RETURN
6750  REM ——————————PRINT I
6760  HLIN X,X + 2 AT Y
6770  HLIN X,X + 2 AT Y + 6
6780  VLIN Y,Y + 6 AT X + 1
6790 X = X - 2
6800  RETURN
6810  REM —————PRINTS J
6820  HLIN X,X + 4 AT Y
6830  VLIN Y,Y + 5 AT X + 3
6840  PLOT X,Y + 5: HLIN X + 1,X + 2 AT Y + 6
6850  RETURN
6860  REM ————————PRINTS K
6870  VLIN Y,Y + 6 AT X
6880  PLOT X + 3,Y + 1: PLOT X + 2,Y + 2: PLOT X + 1,Y + 3: PLOT X + 1,Y +
     4: PLOT X + 2,Y + 5: PLOT X + 3,Y + 6
6890 X = X - 1: RETURN
6900  REM —————————PRINTS L
6910  VLIN Y,Y + 6 AT X
6920  HLIN X,X + 3 AT Y + 6
6930 X = X - 1: RETURN
6940  RETURN
6950  REM —————————PRINTS M
6960  VLIN Y,Y + 6 AT X
6970  VLIN Y,Y + 6 AT X + 4
6980  PLOT X + 1,Y + 1: PLOT X + 3,Y + 1
6990  PLOT X + 2,Y + 2
7000  RETURN
7010  REM —————————PRINTS N
7020  VLIN Y,Y + 6 AT X
7030  VLIN Y,Y + 6 AT X + 4
7040  PLOT X + 1,Y + 1: PLOT X + 2,Y + 2: PLOT X + 2,Y + 3: PLOT X + 3,Y +
     4
```

FINANCIAL MANAGEMENT SYSTEM II

A FAST, EASY-TO-USE ACCOUNTING SYSTEM
DESIGNED FOR
HOME AND BUSINESS ACCOUNTING

OBJECTIVE: Enter an entire month's checking, charge card, and cash accounts in just a few minutes using your own personalized macro lists. Instant error correction on all entries. Audit all files by Code and month. PERFECT FOR TAX ACCOUNTING. Powerful new BUDGET MANAGER for planning and comparing budget with audits. Printer routines for listing disk files, balance, reconcile, search, macro lists, audit and budget reports.

ALL THE ORIGINAL FEATURES + NEW BUDGET MANAGER
* 1-3 KEYSTROKE ENTRYS
* AUTOMATIC TAX CODING
* SINGLE OR DUAL DISK DRIVE

★ ACCOUNT MANAGER: A self-prompting, error avoiding entry system which includes disk files, balance, reconcile, edit, and sort.
★ BUDGET MANAGER: Plan, review, and balance your budget. Then generate complete reports with summation for any 1 - 12 month period.
★ SYSTEM UTILITY: Enter your own Item and tax Code Macros, up to 100 each. Configure program to match almost any printer/disk system.
★ SEARCH RECORDS: Search for any given data. Make specific and expanded searches using the Macro lists.
★ ACCOUNT AUDITOR: Totals all files by tax Code and any 1-12 month period with year-to-date totals.
★ 48K APPLE with ROM APPLESOFT and disk required (printer optional)

PRICE: $39.95 -- Check, VISA, or MASTER CHARGE accepted.

D R JARVIS COMPUTING
1039 Cadiz Dr.- Simi, CA 93065
Phone (805) 526-0151

Dealer Inquiries Invited

START

Educational Software Directory

APPLE II EDITION

APPLE COMPUTER INC. says, in announcing their *Apple Seed Computer Literacy Kit for Schools: "EDUCATIONAL SOFTWARE DIRECTORY,* another new publication from STERLING SWIFT, is the comprehensive bibliography of educational software for **Apple Computers. It** indexes software by grade level and provides short descriptions, sources and prices of the software package."

SS SS

RESPONSE HAS BEEN OVERWHELMING. Please order quickly, directly from Sterling Swift Publishing Company.
Dealer price $11.95, less usual discount
Educational price $9.95
Add $1 for postage & handling; Texas residents add 4% sales tax
Name ______
Address (school, if applicable)(street, city, state, zip) ______
______
______
______

STERLING SWIFT PUBLISHING COMPANY
P. O. Box 188, Manchaca, TX 78652–(512) 444-7570

BOX 3435 • LONGWOOD, FL 32750 • (305)862-6917

WHY ANOTHER VERSION OF ADVENTURE?

Since Will Crowther and Don Woods created it years ago, ADVENTURE has been programmed to run on nearly every computer known to man. The original Fortran version ran on a large PDP machine requiring nearly 300K of storage. At least three other versions of ADVENTURE exist for the APPLE. Some claim to be the only complete version; some claim to fill whole disks with program and data. All, however, require diskette access during the game to retrieve text for display on nearly every command. All added "features" of their own, changed some of the original text, or omitted something from the original.

In this version of ADVENTURE you get nothing but the real thing. NOTHING has been added or left out. By using a text compression technique seldom used on microcomputers, the nearly 44K bytes of text fit in less than 25K. This means no disk access is needed during the game and that it can be played on 48K systems without disk drives. Both tape and Disk versions are identical, only the media is different. The tape version includes instructions for making a disk copy. All versions work with either 3.2 or 3.3 APPLE disks and APPLE 2 or APPLE 2 Plus machines.

**ALSO INCLUDES
COMPLETE SAVE & RESTORE GAME FEATURES**

CLASSIC ADVENTURE
First of the "Other Venture" Series
by the Company that brought you ADVENTURE by Scott Adams

48K APPLE 2 - Tape to Disk	**041-0106**	**$14.95***
48K APPLE 2 - Disk	**042-0106**	**$20.95***

PRICES SHOWN SUBJECT TO CHANGE
*Plus shipping and handling

AVAILABLE FROM YOUR COMPUTER STORE OR ORDER TOLL FREE FROM A.I.

ai Adventure INTERNATIONAL
A Division of Scott Adams, Inc.

ORDER
TOLL FREE
800-327-7172
in FLA. call 305 862-6917
FREE CATALOG

```
7050  PLOT X + 3,Y + 5
7060  RETURN
7070  REM ---------------PRINTS O
7080  VLIN Y,Y + 6 AT X
7090  HLIN X,X + 3 AT Y
7100  HLIN X,X + 3 AT Y + 6
7110  VLIN Y,Y + 6 AT X + 3
7120 X = X - 1: RETURN
7130  REM ---------------PRINTS P
7140  VLIN Y,Y + 6 AT X
7150  HLIN X,X + 3 AT Y
7160  HLIN X,X + 3 AT Y + 3
7170  VLIN Y,Y + 3 AT X + 3
7180 X = X - 1: RETURN
7190  REM ---------------PRINTS Q
7200  VLIN Y,Y + 6 AT X
7210  VLIN Y,Y + 6 AT X + 4
7220  HLIN X,X + 4 AT Y
7230  HLIN X,X + 4 AT Y + 6
7240  PLOT X + 3,Y + 5: PLOT X + 2,Y + 4
7250  PLOT X + 2,Y + 4
7260  RETURN
7270  REM ---------------PRINTS R
7280  VLIN Y,Y + 6 AT X
7290  HLIN X,X + 3 AT Y
7300  HLIN X,X + 3 AT Y + 3
7310  VLIN Y,Y + 3 AT X + 3
7320  PLOT X + 1,Y + 4: PLOT X + 2,Y + 5
7330  PLOT X + 3,Y + 6
7340  IF X > 1 THEN X = X - 1
7350  RETURN
7360  REM ----------------PRINTS S
7370  HLIN X,X + 3 AT Y
7380  HLIN X,X + 3 AT Y + 6
7390  HLIN X,X + 3 AT Y + 3
7400  VLIN Y,Y + 3 AT X
7410  VLIN Y + 3,Y + 6 AT X + 3
7420 X = X - 1
7430  RETURN
7440  REM ---------------PRINTS T
7450  VLIN Y,Y + 6 AT X + 2
7460  HLIN X,X + 4 AT Y
7470  RETURN
7480  REM ----------------PRINTS U
7490  VLIN Y,Y + 6 AT X
7500  VLIN Y,Y + 6 AT X + 3
7510  HLIN X,X + 3 AT Y + 6
7520 X = X - 1: RETURN
7530  REM ----------------PRINTS V
7540  VLIN Y,Y + 3 AT X
7550  VLIN Y,Y + 3 AT X + 4
7560  VLIN Y + 4,Y + 5 AT X + 1: VLIN Y + 4,Y + 5 AT X + 3
7570  PLOT X + 2,Y + 6
7580  RETURN
7590  REM ---------------PRINTS W
7600  VLIN Y,Y + 5 AT X
7610  VLIN Y,Y + 5 AT X + 4
7620  PLOT X + 1,Y + 6: PLOT X + 3,Y + 6
7630  VLIN Y + 4,Y + 5 AT X + 2
7640  RETURN
7650  REM ---------------PRINTS X
7660  VLIN Y,Y + 1 AT X: VLIN Y,Y + 1 AT X + 4
7670  VLIN Y + 5,Y + 6 AT X: VLIN Y + 5,Y + 6 AT X + 4
7680  PLOT X + 1,Y + 2: PLOT X + 3,Y + 2: PLOT X + 1,Y + 4: PLOT X +
     4: PLOT X + 2,Y + 3
7690  RETURN
7700  REM ---------------PRINTS Y
7710  VLIN Y,Y + 1 AT X: VLIN Y,Y + 1 AT X + 4
7720  PLOT X + 1,Y + 2: PLOT X + 3,Y + 2
7730  VLIN Y + 3,Y + 6 AT X + 2
7740  RETURN
7750  REM ----------------PRINTS Z
7760  HLIN X,X + 4 AT Y
7770  HLIN X,X + 4 AT Y + 6
7780  PLOT X + 4,Y + 1: PLOT X + 3,Y + 2: PLOT X + 2,Y + 3: PLOT X +
     4: VLIN Y + 5,Y + 6 AT X
8000  REM --PUNCTUATION ROUTINES-
8010  REM ---- QUESTION MARK---
8020  PLOT X,Y + 1: HLIN X + 1,X + 3 AT Y
8030  VLIN Y + 1,Y + 2 AT X + 4
8040  HLIN X + 2,X + 3 AT Y + 3
8050  PLOT X + 2,Y + 4: PLOT X + 2,Y + 6
8060 X = X + 6
8070  RETURN
8080  REM --- EXCLAMATION POINT
8090  VLIN Y,Y + 4 AT X + 1
8100  PLOT X + 1,Y + 6
8110 X = X + 3
8120  RETURN
8130  REM ----- COMMA -----
8140  VLIN Y + 5,Y + 6 AT X
8150  VLIN Y + 5,Y + 7 AT X + 1
8160 X = X + 4
8170  RETURN
8180  REM ----- PERIOD -----
8190  PLOT X,Y + 6
8200 X = X + 3
8210  RETURN
8220  RETURN
8230  REM ---SINGLE QUOTES---
8240  VLIN Y,Y + 1 AT X
8250 X = X + 2: RETURN
8260  END
8270  REM -----SEMICOLON-----
8280  VLIN Y + 2,Y + 3 AT X
8290  VLIN Y + 2,Y + 3 AT X + 1
8300  VLIN Y + 5,Y + 6 AT X
8310  VLIN Y + 5,Y + 7 AT X + 1
8320 X = X + 4
8330  RETURN
8340  END
```

©

Computer House Division

PROGRAMS FOR COMMODORE AND APPLE

Legal accounting Demo	$15.00
Legal accounting Program	995.00
Machine Part Quote Demo	15.00
Machine Part Quote Program	325.00
Mailing/phone list	80.00
Political Mail/phone list	130.00
Beams, structural	115.00
Trig/Circle Tangent	110.00
Spur Gears	35.00
Bolt Circles	25.00
Filament Wound TAnks	125.00
Scrunch	25.00

PROGRAMS FOR COMMODORE ONLY

A/P, A/R, Job Cost & Job Est.	370.00
Inventory	95.00
Financial	175.00
Real Estate Listings	265.00
Check Writer	25.00
File Editing Tools (FET)	65.00
Screen Dump/Repeat	35.00
Docu-Print	20.00
Scrunch	25.00
Sof-Bkup	40.00
Sorter (Mach. Language)	35.00
Trace-Print	25.00
Vari-Print	25.00

ASK FOR CATALOG #80-C2 Dealers Wanted
Computer House Div. 1407 Clinton Road
Jackson, Michigan 49202 (517) 782-2132

THE PERFORMANCE SLICE

SBCS

GENERAL LEDGER

This package features 31 character account names, 6 digit account numbers, and 10 levels of subtotals for more detailed income statements and balance sheets. Up to 2000 entries can be processed per session.

ACCOUNTS RECEIVABLE

This package allows entry of invoices at any time, credit and debit memos, full or partial invoice payment, invoice aging, and printing of statements. Amounts billed this year and year previous, total payments, and progress billing information are maintained.

ACCOUNTS PAYABLE COMING SOON!

★ Complete your accounting system with the soon to be released A/P package, featuring automatic application of credit and debit memos, open or closed item listing, full invoice aging, and multiple reports that provide a complete transaction review.

★ **Your bookkeeping doesn't have to be a bulky, complicated process. The SBCS Accounting System is designed for flexibility and high performance with a cost effectiveness sure to benefit your business!**

YOU NEED EXPERIENCE WORKING FOR YOU

★ **Packages available at your local Apple dealer.**

SMALL BUSINESS COMPUTER SYSTEMS
4140 Greenwood, Lincoln, NE 68504 (402) 467-1878

At the 'CORE' of every Apple should be a FURN-WOOD DESK ...

Quality wood furniture designed specifically for Apple, Atari and TRS-80 computers. Many configurations to choose from at low cost. All priced to move! Send for complete details, prices and brochures.

FURN-WOOD MANUFACTURING, INC.

5664 S.W. Carman Drive • Lake Oswego, Oregon 97034
(503) 636-1991

COMPUTE!'s Book Corner

Back Issues:

COMPUTE! January, 1981, Issue 8	$2.50	☐
COMPUTE! February, 1981, Issue 9	$2.50	☐
COMPUTE! March, 1981, Issue 10	$2.50	☐

For The Single Board Computer Owners: (KIM, SYM, AIM, OSI)

COMPUTE II, All Three Issues
(The only ones: **COMPUTE II** merged into **COMPUTE!**)
Issues 1, 2, and 3 .... $5.00 ☐

Individual Issues

COMPUTE II, #1	$2.00	☐
COMPUTE II, #2	$2.00	☐
COMPUTE II, #3	$2.00	☐

Miscellaneous:

Best of the PET Gazette .... $10.00 ☐
(Note...an old anthology primarily for old ROM, 8K original PETs.)
Collected PET User Notes (7 issues) .... $10.00 ☐
(Above note applies)
Collected 6502 User Notes (17 issues) .... $20.00 ☐
(Specifically for Single-Board Computer Owners)

Ordering Information:

Address orders to:
Compute's Book Corner
P.O. Box 5406
Greensboro, NC 27403 USA

Payment, in US funds, required with order. *Master Charge/VISA accepted. All items subject to availability. Please add the following amount(s) for shipping/handling:
US $2.00 (UPS or mail) • Canada $2.00 (mail)
Foreign Air Mail $5.00

OUR NEW PROGRAM PACKAGE
FOR THE APPLE II
LOOKS TERRIFIC

SO DOES THE PRICE

Can 25 quality Apple programs cost a mere $24.95?

We have a lot to gain by almost losing our shirts. Because we are a new and growing company, we need your business to build our business.

APPLE SACK I — Twenty-five Applesoft programs with everything from hi-res graphics to adventures. Programs include 5 business, 5 utility, 10 games and 5 home programs.

APPLE SACK II — TV TRIVIA — Hundreds of questions about your old favorite TV shows from the 50's and the 60's. Extra points given for super stumpers. Will tingle your memory cells. (Integer Basic)

APPLE SACK III — ADVENTURES — 8 Adventures for excitement in your life. Travel to far away places, thru space, into caves, explore dungeons and fight for your life.

EACH PACK IS ONLY $24.95 AND INCLUDES SHIPPING — GET ALL THREE PACKAGES FOR $49.95.

It's not that our program packages for the Apple are so cheap. It's just that the other companies are so expensive.

ORDER YOUR SACK TODAY FROM . . . .

COMPUTERMAT
Box 1664C
Lake Havasu City, Arizona 86403
(602) 855-3357

Decrementing The For... Next & Endless Loops

Derek Kelly

Scrolling text *backward* in a RUNning program can be programmed almost as easily as scrolling text forward. Scrolling text forward can be easily accomplished, as can be seen in the program in figure 1:

```
10 For I = 1 to 20
20 Print A$(I)
30 Get G$
40 Next I.
```

The GET command in Applesoft BASIC allows a pause-until-any-key-is-hit way for a user to control the progress of a program. Here, after every string printed (A$(I), a pause is inserted. After each pause, the printing of strings will proceed in a sequential manner, incrementing the counter for 'I' by 1 each time. This is forward scrolling.

But what if you want to go backward? Suppose you want the previous string by a count of 1, 2, to the first printed string? Can you go backward? The answer is Yes, and it's simple to accomplish. But like the various problems associated with For... Next loops generally, if not properly constructed, e.g., with no GOTO's that branch away totally from the loop, is that For... Next loops can involve endless loops, whether such loops run forward or backward.

In any well-constructed rogram module, there should be only one point of entrance to that module, and only one real exit. Some programmers who use GOTO, and related statements, have been known to construct programs where *under normal use* the program will "hang" somewhere, a somewhere often caused by a GOTO or a number of GOTO's which send the program around in circles. For example, consider programs A and B below. A has one entrance and exit, B has many:

```
 10 GOSUB 100
  .
  .
100 For I = 1 to 230: Print "!";: GOSUB 5
110 GOSUB 200: GOSUB 300
120 NEXT I
130 RETURN
```

Program A

```
10 For I = 1 to 33
20 If I = Int(I/11) Then GOTO 101
30 GET G$: If G$ = "?" Then GOTO
   163
 .
 .
 .
163 Print " = ": GOTO 194
101 For J = 1 To 3: If I = J
    Then GOTO 121: Next J
102 GOTO 30
```

Program B

Program A is called by a GOSUB from line 10. Program A does one set of functions in the program as a whole. When called, it does what it's supposed to, and then returns. Program B, on the other hand, GOTO's all over the place, and one can't predict that any one GOTO will even find its way back to the place that started the GOingTO process. Under normal conditions, and presupposing that the GOSUBs called by Program A are well-constructed, then we can reasonably predict that program A will not cause an endless loop, while Program B most certainly *may* contain an endless loop or two that will "hang" your computer.

The two short programs above affect the forward running of For... Next loops. If a user desires to decrement rather than increment a For... Next loop so as to reverse the order of a printout, and be able to review previously printed items, then other endless loop possibilities arise.

Take Program A as an example. I might want to add a pause to the program, so I add III Get G$: If G$ = "?" Then GOTO 130. This GOTO is OK because it takes place *within* the routine.

Now I want to add a provision to reverse the loop. Since the "NEXT I" statement increments a counter by 1 (or whatever), to reverse print, I must subtract 2 from the I counter to get the next previous record. So my program now looks like PROGRAM C:

```
100 For I = 1 to 230: Print "!";: GOSUB 5
110 GOSUB200 : GOSUB300)
111 Get G$: If G$ = "-" Then I = I-2
120 NEXT I
130 RETURN
```

Program C

As soon as Program C is RUN, you will see that it decrements to 1 then to -1 and so on. You don't want negative numbers, so another line of code will have to be added. In addition, when the counter gets to '-1', the computer — at least *my* APPLE — hangs and prints out a steady, unstoppable — except by RESET, stream of 1's...1.1.1.1.1.1.1.1.1.1.1....
This problem can be avoided by changing line 111 to read:

111 Get G$: IF G$ = "-" Then I = I-2: If

PROGRESSIVE SOFTWARE

Presents

Software and Hardware for your APPLE

VU #1

- Accepts data from VISICALC*
- Relocates data and names
- Connects data and names
- Creates an out file that can be loaded into VISICALC*

Written by: MARC GOLDFARB $49.95
Add $4.95 for Postage and Handling.

VU #2

Includes all of the features of VU #1. VU #2 is a utility program for VISICALC* that has a built-in execute file that operates the program automatically and allows the user to do repetitious functions, such as payroll computing, etc. Written by:
MARC GOLDFARB $59.95
Add $4.95 for Postage and Handling.

VU #3

Includes the abilities of VU #1 and VU #2 but has additional superb features. This program will allow the user to enter data into VISICALC* from any program merely by inserting data into an array (which is well documented in the instructions). Then the program places the array into VISICALC*. VU #3 will also transfer data generated from VISICALC* into any of the user's programs via an array (as defined in the instructions). Written by:
MARC GOLDFARB $69.95
Add $4.95 for Postage and Handling.

Menu Planner — Has many types of dishes, counts calories for each entry. Prints out or displays menus with daily calorie count. Written by:
GARY A. FOOTE $19.95

Shopping List — Creates a list that can be used for food shopping or any other group of items. Allows user to create and save files of other items Prints out list. Written by:
CHARLES HEARN $19.95

Apple Menu Cookbook — Index-accessed data storage/retrieval program. Recipes stored, unlimited lines per entry. Easy editing. Formulated after NY Times Cookbook. Other useful features included. Written by:
WM. MERLINO, MD $19.95

Data Organizer — Alphabetize and maintain any data list up to 1,000 entries. Searching feature. Written by:
ED HOWERTON $14.95

*VISICALC is a trade-mark of Personal Software, Inc.

- Programs accepted for publication.
- Highest royalty paid.

FILES

- Builds Serial Files
- Changes Serial Files to Random Access Files
- Adds to End of Serial Files
- Record Insertion and Deletion Anywhere in Serial File
- Move Individual Records or Blocks of Records within Serial Files

A File manipulator that allows the user to search for a string within a file, sort data by blocks, handle many files at a time (without exiting the program) and saves executed files. A file can be saved under many names, viewed in several modes, and dumped (totally or partially) to a printer.

If you would like to work with files you must own this program! Comes with 10-page documentation in a binder. Because of the size and weight of this program, postage and handling charge is necessary. Written by:
MARC GOLDFARB FILES-Disk only. $49.95
Add $4.95 for Postage and Handling.

Super Sort and Search — A free-form data base package which allows the user to define fields per record. As the data is loaded in, it is automatically sorted. Can also merge together files or allow key sort entries. Will sort out in particular field, do a generic search and/or search for a specific piece of data. Will allow each particular record or records to be printed. Written by:
MARC GOLDFARB $69.95
Add $4.95 for Postage and Handling.

Matching Patterns — Choose two colored patterns that are exactly the same. Music response on correct patterns. Higher degree of accuracy produces longer melody before changing to more complex pattern (4 levels). Increases awareness of contrast and comparison. Written by:
JUDY PEGG Disk $19.95

Memorex Diskettes — Box of 10 - $24.00. Quantity Orders call for Discount. (Apple, Pet, TRS-80 and Ohio Scientific).

TO ORDER

Send Check or Money Order to:

P. O. Box 273
Plymouth Meeting, PA 19462

Pennsylvania residents add 6% sales tax.

POSTAGE AND HANDLING

Please add $1.50 for the first item and $1.00 for each additional item.
U.S. and foreign dealer and distributor inquiries invited.
All programs require 16K memory unless specified.

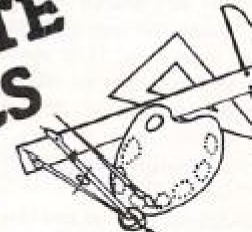
215-279-4438

I< = 0 Then I = 0 If I = 0 or less than zero, then the first item will be printed, as I will = 0 *before* the execution of the NEXT which will increment I by 1 (or more) to 1.

The simple program C, if improved to include the line 111 above, will be able to process the forward and backward scrolling in any program in which it is called.

In general, there is no formula that can guarantee that a program has no endless loops. It is always possible for some bug such as an endless loop to exist under certain conditions in any program. Since you can't be sure about the absence of bugs, you have to make do with the presence of controls to limit the possibility of harmful bugs. ©

THE COMPLETE GRAPHICS SYSTEM



by Mark Pelczarski
for the Apple II

- Over 100 Hi-Res colors!
- Drawing program includes automatic filling and user-definable "paintbrushes".
- The finest 3-D graphics available, capable of color and multiple figures with machine language speed.
- Create shape tables with paddles or keystrokes.
- Hi-Res text anywhere on the screen. Define your own fonts.
- Thorough, easy-to-use manual even shows how to use drawings and 100 color fill routine in your own programs.

$59.95 on disk
DOS 3.2 or 3.3, requires Applesoft firmware or language system.

from **CO-OP software**
a division of Micro Co-op

VISA and MasterCard accepted
Dealer inquiries invited

Post Office Box 432
West Chicago, IL 60185
(312) 231-0912

Apple II is a trademark of Apple Computer, Inc.

HUNTINGTON COMPUTING

PROGRAMS LISTED BELOW ARE ON DISK:

Memorex Disks (Special til 4-15-81) **Ten for**		**$34.99**
Apple Galaxian	$24.95 now	**$32.99**
EPYX (Automated Simulations) SPECIAL — While they last: Ryn/Morloc/Rigel all three		**$39.95**
Galaxy Wars	$24.95 now	**$22.99**
Super Script	$89.95 now	**$76.39**
Interactive Fiction	$14.95 now	**$12.99**
Alien Lander	$24.95 now	**$21.99**
Sorcerer/Doom Cavern	$20.00 now	**$17.99**
Hyper Head-On	$24.95 now	**$21.20**
Galactic Empire	$24.95 now	**$21.20**
Galactic Trader	$24.95 now	**$21.20**
Galactic Revolution	$24.95 now	**$21.20**
Golden Mountain	$19.95 now	**$16.95**
Tank Command	$14.95 now	**$12.49**
DB Master	$189.00 now	**$169.00**
Bloody Murder	$19.95 now	**$17.99**
Tranquility Base	$24.95 now	**$22.99**
Aristotle's Apple	$34.95 now	**$29.99**
Micro Memo	$39.95 now	**$33.99**
The Data Factory	$150.00 now	**$134.00**
Mission Asteroid	$19.95 now	**$17.99**
Vinyl holders for 20 disks in beautiful deluxe padded binder		**$12.99**
Skybombers II	$19.95 now	**$17.99**
Apple-Doc	$24.95 now	**$21.99**
Programmers Utility	$16.95 now	**$13.89**
DOS Tool Kit	$75.00 now	**$69.00**
Microsoft Adventure	$29.95 now	**$26.89**
Olympic Decathlon	$29.95 now	**$26.89**
Compu-Math: Arithmetic	$49.95 now	**$44.95**
Apple Plot	$75.00 now	**$69.00**
College Boards (Krell)	$79.95 now	**$69.95**
Time Traveler	$24.95 now	**$22.99**
Sword of Zedek	$24.95 now	**$22.99**
All Time Super Star Baseball	$14.95 now	**$13.19**
CRAE 2.0	$24.95 now	**$21.99**
MCAT 2.0	$19.95 now	**$17.99**
The Tarturian	$24.95 now	**$21.99**
TI PROGRAMMER — Hexidecimal/Octal Calculator Retail $65.00	now	**$59.99**
MPI 88G Printer		**$499.00**
The Prisoner	$29.95 now	**$26.99**
The Wizard & The Princess	$32.95 now	**$28.99**
Compu-Spell	$39.95 now	**$34.99**
Computer Ambush	$59.95 now	**$52.99**
Computer Bismark	$59.95 now	**$52.91**
Computer Napoleonics	$59.95 now	**$52.99**
Computer Quarterback	$39.95 now	**$33.99**
Flight Simulator (disk)	$34.95 now	**$29.50**
Star Cruiser	$24.95 now	**$20.99**
Space Album	$39.95 now	**$34.50**
Odyssey	$29.95 now	**$25.50**
Both Barrels	$24.95 now	**$20.99**
Modifiable Database	$79.50 now	**$64.99**
Micro League Baseball	$24.95 now	**$22.99**
Sargon II	$34.95 now	**$29.70**
Program Line Editor	$40.00 now	**$34.99**
Retro-Ball	$29.95 now	**$25.99**
Space Eggs	$29.95 now	**$25.99**
Mystery House in French		**$22.99**
Z-80 Softcard with CP/M	$349.00 now	**$299.00**
Videx 80-Col. Board	$350.00 now	**$299.00**
EZ Draw	$34.95 now	**$29.99**
Head On	$24.95 now	**$20.99**
3D Super Graphics	$39.95 now	**$33.99**
Compu-Math I or II	$40.00 now	**$34** ea.
HI-RES Cribbage	$24.95 now	**$21.99**
Phantom Five	$29.95 now	**$25.49**
Star Gazer's Guide	$30.00 now	**$25.39**
Lords of Karma	$20.00 now	**$16.99**
Tawala's Last Redoubt	$29.95 now	**$25.49**
Apple PIE & Formatter (Reg. $129.95)	Special	**$99.99**
Apple World	$59.95 now	**$52.99**
ABM (Muse)	$24.95 now	**$22.49**
Data Plot (Muse)	$59.95 now	**$52.99**
Computer Conflict	$39.95 now	**$33.99**
Computer Air Combat	$59.95 now	**$52.99**
Terrorist	$29.95 now	**$25.45**
The Temple of Apshai	$39.95 now	**$33.95**
Super-Text II	$150.00 now	**$129.92**
Magic Window	$99.95 now	**$85.00**
CCA Data Management	$99.95 now	**$85.00**
VISICALC	$150.00 now	**$119.00**
Acant. Fortune-telling	$15.00 now	**$8.99**
Horriblescope	$15.00 now	**$8.99**
Gomoku (cass.)	$14.95 now	**$11.99**
Reversal	$34.95 now	**$29.70**
16K NEC Memory		**$49.99**
Savage Island Adv. (cass.)	$14.95 now	**$12.70**
Baker's Trilogy	$24.95 now	**$21.99**
Crosswords	$29.95 now	**$25.99**
Microgammon II	$19.95 now	**$17.99**
Pensoft	$29.95 now	**$25.99**
Dogfight	$29.95 now	**$25.99**
440G Graphics	$34.95 now	**$29.99**
460G Graphics	$34.95 now	**$29.99**
Cyber Strike	$39.95 now	**$34.99**
HI-RES Football (On-Line)	$39.95 now	**$34.99**
Fastgammon	$24.95 now	**$21.30**
The Voice	$39.95 now	**$33.99**
Hellfire Warrior	$29.95 now	**$25.45**
Beneath Apple Manor	$20.00 now	**$17.45**
Astro Apple	$20.00 now	**$17.45**
Akalabeth	$34.95 now	**$29.99**
ANDROMEDA 16K RAM Expansion Board for Apple II Retail is $195 — Our price		**$184.99**
NEC 12" Green/Black	$260 now	**$219.00**
Centronics 730 Printer	$795 now	**$669.00**
Centronics 737 Printer	$995 now	**$799.00**
ABT Pad for Apple	$125 now	**$109.00**
ROM - w/filter	$200 now	**$170.00**
M&R 80-col. Video Card	$375 now	**$335.00**
Versawriter	$249 now	**$219.00**

Send for TRS-80, PET and ATARI Catalog also.

apple SOFTWARE

SPECIAL up to 40% OFF LIST

FASTEST SERVICE

We take MasterCard or VISA (include card # and expiration date). California residents add 6% tax. Include $2.00 for postage. $5.00 Foreign. Mail to

HUNTINGTON COMPUTING, Dept. COM 3
P. O. Box 787, 2020 Charles, Corcoran, CA 93212
24-hour order service. Call
(209) 992-5411. Send for free catalog.

HELP WANTED

Apple Authors

COMPUTE! is looking for applications articles aimed at beginners and intermediate programmers. We're specifically interested in programming hints, tutorials, articles written to help users get more out of their machine. We're not afraid of re-inventing the wheel (those useful programming tricks you learned two years ago are just as relevent to a beginner now), but we do insist on high quality, pertinant information. We pay on a published page basis, and that includes program listings as well. Here are our brief guidelines: Text should be typed in upper/lower case, double spaced, with your name and name of the article appearing at the top of each page. Program listings should be single spaced, with a new ribbon on your printer.

Apple User Groups

Please take the time to announce this at your next meeting. COMPUTE! is serious about supporting the Apple with the same high quality information we provide for other 6502 machines. To do that, we need your support as well.

Mailing address: COMPUTE!
Post Office Box 5406
Greensboro, NC 27403 USA
Telephone: (919) 275-9809

Got A Question You'd Like Answered?

Write: Ask The Readers,
c/o **COMPUTE!**,
P.O. Box 5406,
Greensboro, NC 27403

APPLE II
SOFTWARE FROM
POWERSOFT
P. O. BOX 157
PITMAN, NEW JERSEY 08071
(609) 589-5500

SUPER CHECKBOOK 3.0—A **vastly improved** version of our popular selling program. With new features such as: simplified but powerful transaction entry and modification routines, new reconciliation routines, additional features such as 30 percent increase in the total number of checks handled, posting of interest from interest bearing checking accounts, automatic teller transactions, bullet proof error handling, and smart disk routines. Plus the program still contains the options of bar graphs, sorting, activities, and account status.
Disk Only/Applesoft $34.95

Of special interest to owners of older versions of the program is a limited trade-in period ending May 31, 1981. Trade-in value $19.95 cassette $24.95 diskette. Original tape or diskette must be returned to receive new version plus $1.50 freight and the difference between $34.95 and the trade-in value. A conversion program is included to convert data files to the new format.

ADDRESS FILE GENERATOR—Allows the user to create any number of four types of address files: Holiday, Birthday, Home and Commercial. The program contains a menu of seven major commands used to: Create, Add, Edit, Display, Search, Sort, and Reorganize Files. Up to three fields may be used for the sort criteria. Major commands have subordinate commands which adds to the flexibility of this powerful software system. We doubt you could buy a better program for maintaining and printing address files .... Disk Only/
Apple Printer Card/Applesoft $24.95

SPANISH VOCABULARY DRILL
FRENCH VOCABULARY DRILL
ITALIAN VOCABULARY DRILL
GERMAN VOCABULARY DRILL

These programs provide practice in foreign language vocabulary by means of three types of drills: Matching, **Foreign Language to English and English to Foreign Language.** Although the diskette comes with some lessons on it, these are intended to be samples. The most effective way to use these programs is to enter your own lessons from the course you are studying. To facilitate the entry of new lessons, each program contains a complete Lesson Editor which has various entry and revision options. The manual also contains instructions for converting the programs to other languages .... Disk Only/
Applesoft each $24.95

SPACE TREK J—Your mission is to patrol the galaxy, and to seek out and destroy the ships of the Klarian fleet. At your command is the starship Lexington. The Lexington has a wide variety of weapons, sensors and other devices useful in your mission. There are two kinds of Klarian ships Regular and Super. Regular Klarians have an average energy supply of 5000 to 12000 quarks while Super Klarians have 12500 to 15000 quarks and are protected from some of the Lexingtons' weapons and attack strategies .... Disk Only/
Applesoft $19.95

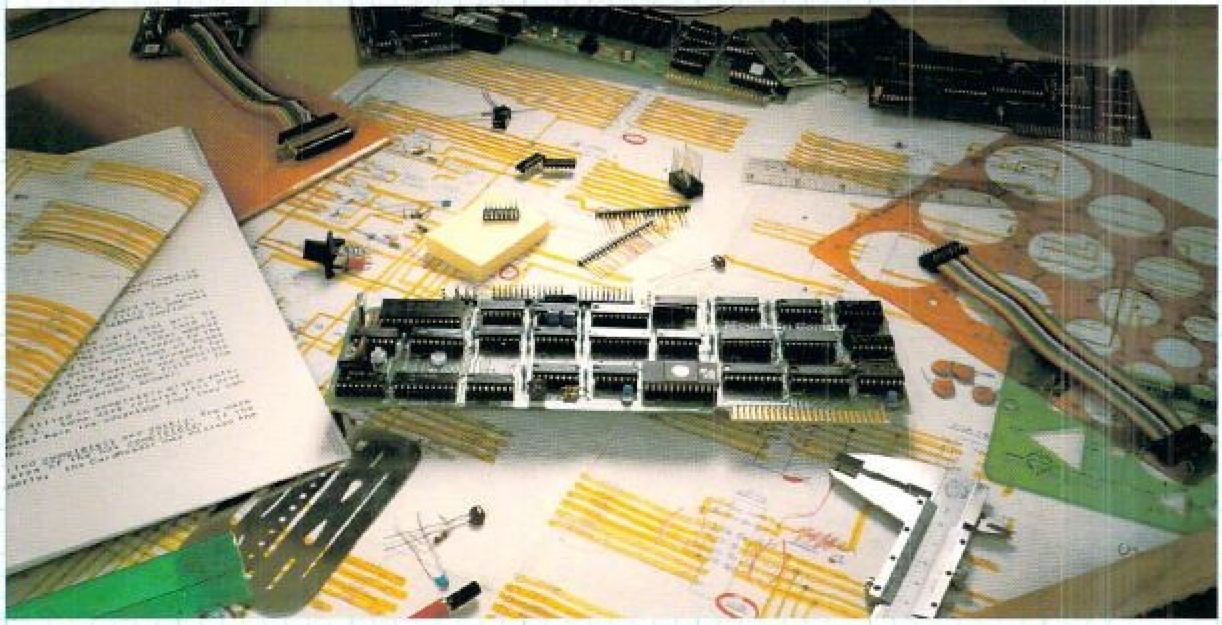
WORLD OF ODYSSEY—An adventure game utilizing the full power of Disk II, which enables the player to explore 353 rooms on 6 different levels full of dragons, dwarfs, orcs, goblins, gold and jewels. The program allows the player to stop the game and to resume at a later time .... Disk Only/Applesoft $24.95

GALACTIC EMPIRES—Pits 1 to 20 players against each other and the computer in a struggle for control of up to 40 star systems. The players compete by sending out fleets of ships to capture neutral planets and to attack the colonies of other players. The victor is the player who controls the most stars by game's end.
Applesoft $14.95

Dealer Inquiries Invited—Visa and MasterCard

POWERSOFT
P. O. BOX 157
PITMAN, NEW JERSEY 08071
(609) 589-5500

Mountain Computer put it all together for you.



The CPS MultiFunction Card

Three cards in one! The Mountain Computer CPS MultiFunction Card provides all the capabilities of a serial interface, parallel output interface and real-time clock/calendar—all on one card—occupying only one slot in your Apple II®. Serial and Parallel output may be used simultaneously from CPS. CPS is configured from a set-up program on diskette which sets the parameters (such as baud rate, etc.) for all functions contained on the card and is stored in CMOS RAM on the card. Once you have configured your card, you need never set it up again. You may also change parameters from the keyboard with control commands. All function set-ups stored on-board are battery powered for up to two years. "Phantom slot" capability permits assigning each of the functions of CPS to different slots in your Apple without the card actually being in those slots! For example, insert CPS in slot #4 and set it up so that is simulates a parallel interface in slot #1 and a clock in slot #7 and leave the serial port assigned to slot #4. CPS's on-board intelligence lets it function in a wide variety of configurations, thereby providing software compatibility with most existing programs. "We've put it all together for you"—for these reasons and many more! *Drop by your Apple dealer and see for yourself how our CPS MultiFunction Card can expand the capabilities of your Apple and save you a great deal of money as well!*

Calendar/Clock
- One second to 99 years
- Battery backed-up (2 years)
- Two AA standard alkaline batteries for back-up (provided)
- Compatible with MCI Apple Clock™ time access programs

Parallel Output
- Features auto-line feed, Apple tabbing, line length, delay after carriage return, lower to upper case conversion
- Centronics standard—reconfigurable to other standards
- Status bit handshaking

Serial Interface
- Features auto-line feed, transparent terminal mode, Apple tabbing, line length, delay after carriage return, local echo of output characters, simultaneous serial/parallel output, lower to upper case conversion, discarding of extraneous LFs from serial input
- Uses the powerful 2651 serial PCI chip
- 16 selectable internal baud rates—50 to 19.2Kbaud
- Half/Full duplex terminal operation
- I/O interface conforms to RS-232C
- Asynchronous/Synchronous operation

Mountain Computer
INCORPORATED

300 El Pueblo Scotts Valley, CA 95066
(408) 438-6650 TWX: 910 598-4504

™Apple Clock was the trademark of Mountain Computer Inc.

®Apple and Apple II are registered trademarks of Apple Computer Inc.

Using The Aim 65 As A Remote Terminal For An Apple

Tony Davis and Marvin L. De Jong
Department of Mathematics-Physics
The School of the Ozarks
Pt. Lookout, MO 65726

In the March issue of **COMPUTE!** (page 28 – Computer Communications Experiments) a circuit using the 6551 ACIA (Asynchronous Communications Interface Adapter) and a RS-232C interface to a modem were described. We have used this same interface to a NOVATION CAT modem on the AIM 65 to operate an Apple II over a telephone link. The Apple was equipped with a Hayes micromodem. The Apple was used to run BASIC programs, but its monitor can also be used to load machine language programs or data.

The circuit will not be repeated here, but we will provide the listing of the simple program that we used on the AIM 65. The Hayes Micromodem comes with its own firmware.

We operated the 6551 in the mode where a received character produces an interrupt. The interrupt routine simply prints the character on the display by jumping to an AIM 65 monitor subroutine. The program runs at 300 or 110 Baud. In Listing 1 we show the 6551 initialized to run at 300 Baud. Note that in either case the AIM 65 thermal printer was not used because its print time is so long that several characters are missed. To use it one would have to write a routine to buffer the incoming data. Our ultimate goal is to use the AIM 65 to access the college's big IBM mainframe. I am especially interested in being able to calculate my own salary and print my own paycheck at the end of each month.

Listing 1. Program to operate an Apple from an AIM 65 over a telephone line.

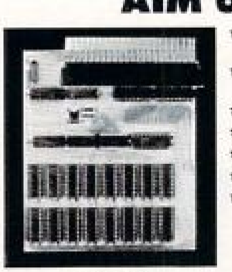
```
$0F00 58          START   CLI            Allow interrupts.
 0F01 D8                  CLD
 0F02 A9  09              LDA  #09       Set up the 6551
                                         command register.
 0F04 8D  02  94          STA  CMNDREG
 0F07 A9  16              LDA  #$13      Set up the control
                                         register for
 0F09 8D  03  94          STA  CNTREG    300 Baud.
 0F0C 20  3C  E9  CHAR    JMP  READ      Get character from
                                         AIM keyboard.
 0F0F 8D  00  94          STA  DATA      Output data to the 6551.
 0F12 AD  01  94  CHECK   LDA  STATUS    Check the status register
 0F15 29  10              AND  #$10      Check bit four.
 0F17 FO  F9              BEQ  CHECK     Wait for data to be
                                         transmitted.
 0F19 D0                  BNE  CHAR      Then get another
                                         character.
                          **********
```

Interrupt Routine

```
$0E00 48          IRQ     PHA            Save the accumulator.
 0E01 AD  00  94          LDA  DATA      Get character that
                                         was sent.
 0E04 20  7A  E9          JMP  OUTPUT    Output character to
                                         display.
 0E07 AD  01  94          LDA  STATUS    Clear IRQ flag.
 0E0A 68                  PLA
 0E0B 40                  RTI
```

Be sure to load the interrupt vector $0E00.

AIM 65-8K STATIC MEMORY



★ Plugs directly onto AIM-65 memory expansion blade.
★ Positions neatly under AIM-65 allowing use of available enclosures.
★ Expansion blade provided for further expansion.
★ +5 volts supplied by host AIM-65.
★ 8K memory board draws only 200 ma.
★ Two separately addressable 4K blocks.
★ KIM-1 compatible.

MEM 4:	8K memory board, 4K RAM chips	$109.00
MEM 8:	8K memory board, 8K RAM chips	$169.00
RAM 4:	4K RAM chips to upgrade MEM 4 to 8K	$ 69.00
	Full documentation kit	$ 1.00

SYSTEM PERIPHERALS
P.O. Box 971, Dept. C.
Troy, MI 48099

PET & APPLE II USERS

TINY PASCAL
Plus +
GRAPHICS

The TINY Pascal System turns your APPLE II micro into a 16-bit P-machine. You too can learn the language that is slated to become the successor to BASIC. TINY Pascal offers the following:

* LINE EDITOR to create, modify and maintain source
* COMPILER to produce P-code, the assembly language of the P-machine
* INTERPRETER to execute the compiled P-code (has TRACE)
* Structured programmed constructs: CASE-OF-ELSE, WHILE-DO, IF-THEN-ELSE, REPEAT-UNTIL, FOR-TO/DOWNTO-DO, BEGIN-END, MEM, CONST, VAR ARRAY

Our new TINY Pascal PLUS + provides graphics and other builtin functions: GRAPHICS, PLOT, POINT, TEXT, INKEY, ABS AND SQR. The PET version supports double density plotting on 40 column screen giving 80 x 50 plot positions. The APPLE II version supports LORES and for ROM APPLESOFT owners the HIRES graphics plus other features with: COLOR, HGRAPHICS, HCOLOR, HPLOT, PDL and TONE. For those who do not require graphics capabilities, you may still order our original Tiny Pascal package.

TINY Pascal PLUS + GRAPHICS VERSION-	
PET 32K NEW Roms cassette	$55
PET 32K NEW Roms diskette	$50
APPLE II 32K/48K w/DOS 3.2 or 3.3	$50

TINY Pascal NON-GRAPHICS VERSIONS-	
PET 16K/32K NEW Roms cassette	$40
PET 16K/32K NEW Roms diskette	$35
APPLE II w/ROM Applesoft 32K w/DOS	$35
APPLE II w/RAM Applesoft 48K w/DOS	$35

USER's Manual (refundable with software order)	$10
6502 Assembly Listing of INTERPRETER-graphics	$25
6502 Assembly Listing of INTERPRETER-non graphics	$20

FREE postage in U.S. and CANADA. Orders may be prepaid or by bankcard (include card number and expiration date). Michigan residents include 4% state sales tax. Orders accepted via THE SOURCE - CL0652

master charge THE INTERBANK CARD

VISA

ABACUS SOFTWARE
P. O. Box 7211
Grand Rapids, Michigan 49510

THE apple® GAZETTE

Using Named GOSUB And GOTO Statements In Applesoft Basic

M. R. Smith

Using subroutines greatly improves the readability of a program and makes it easier to debug. However remembering what a particular GOSUB does is often difficult. Was it GOSUB 1000 or GOSUB 2000 that was wanted?

One of the nice features of Integer Apple Basic is its ability to let you give a name as well as a number in GOSUB statements. The following Integer program demonstrates this:

```
10   GOSUB 100
20   SUB1 =100
30   GOSUB SUB1
40   STOP
100  PRINT "HERE" : RETURN
```

Typing this program whilst using Applesoft will lead to the error message "UNDEFINED STATEMENT IN 30".

The purpose of this program is to show how to use names GOSUB and GOTO statements within Applesoft. By loading the short machine language program described in this article, you are able to run the Applesoft program.

```
10   GOSUB 100
20   SUB1 = 100
30   & GOSUB SUB1
40   STOP
100  PRINT "HERE" : RETURN
```

For the murky details of how it works read the section "PROGRAM DESCRIPTION". Otherwise, type in the demonstration BASIC program and type RUN. The program includes a routine to check that the DATA statements have been entered correctly. Once the demo program has run correctly, the machine language program can be saved using BSAVE NAMED.GOSUB,A$300,L$43. To have the program ready for future sessions, simply type BRUN NAMES.GOSUB as the first part of your programming session. This will load and fix the code. It will remain ready but out of your way until you power down.

WARNING: If you use a RENUMBER program to reorder your program statements, you must remember that variables are NOT changed. Therefore your subroutine pointers will not be renumbered; you'll have to do that by hand.

WARNING: The instructions GOSUB and ON. . .GOSUB are entirely different. The machine code given here will not allow the statement ON X & GOSUB FNAME, SNAME.

Machine Language Program Description

The first statement (at $D93E) of the Applesoft Interpreter GOTO subroutine is the reason that Applesoft does not handle GOSUB's and GOTO's in the same manner as Integer Basic. This statement goes and gets an integer number for use within the GOTO. This means that the BASIC statement GOSUB 1000 is okay but N = 1000 : GOSUB N is not allowed as N as a variable.

Now changing these memory locations to cause the next EXPRESSION to be evaluated, rather than the next NUMBER, allows us to use named GOSUB's. To change these actual locations is impossible. Instead the GOSUB and GOTO routines must be relocated lower in memory at $300 (768) where they can be changed. The Apple's ampersand instruction (&) can then be used to make the new commands operate.

Lines 19–25. Set the ampersand vector (&) at $3F5.
Lines 27–32. Check for GOSUB or GOTO tokens after the &.
Lines 34–47. Relocated version of the monitor GOSUB routine. This now calls the new front end of the GOTO routine.
Lines 49–52. New front end to drive the monitor GOTO routine. It jumps into the middle of the old GOTO routine.
Lines 50 and 51 are the actual major changes.

BASIC Program Description.

Lines 20 and 5000–5200. The program first checks that the DATA statements have been correctly entered. Each pair of DATA statements consists of 16 numbers and a checksum which is the previous 16 numbers added together. If this 17th number is not the actual sum of the previus 16 numbers, then an error is indicated. If all the statements are okay, then the code is loaded.

Line 40. Sets the ampersand vector. This is not necessary if the machine code is BRUN into memory but is necessary if the code is BLOADed.
Lines 60–80. These set the subroutine names.
Lines 100–140. Demonstrate the new instructions.
Lines 1000–3020. Demonstration Subroutines.

References

"AMPERSAND-INTERPRETER" by R. M. Mottala in **Nibble #6**, 1980, p27.
"APPLESOFT INTERNAL ENTRY POINTS" by Apple Computer Inc. in **Apple Orchard**, March/April 1980, p12.
"SOME ROUTINES IN APPLESOFT BASIC" by J. Butterfield in **COMPUTE!**, September/October 1980, p68.

```
]RUN
OKAY
OKAY
OKAY
OKAY
OKAY
BLOAD OKAY


THIS IS JOHN
HERE BY A NAMED GOSUB

THIS IS PETE
HERE BY A DIFFERENT NAMED GOSUB

THIS IS PHREDD
HERE BY A NAMED GOTO

BREAK IN 3020
]&FF
```

```
0300- A9 4C 8D F5 03 A9 10 8D
0308- F6 03 A9 03 8D F7 03 60
0310- C9 B0 F0 09 C9 AB F0 1F
0318- A2 10 4C 12 D4 A9 03 20
0320- D6 D3 A5 B9 48 A5 B8 48
0328- A5 76 48 A5 75 48 A9 B0
0330- 48 20 37 03 4C D2 D7 20
0338- B1 00 20 7B DD 20 52 E7
0340- 4C 41 D9
*
 00 00 00 00 00
*
```

Got A Question You'd Like Answered?

Write: Ask The Readers, c/o **COMPUTE!**, P.O. Box 5406, Greensboro, NC 27403

HUNTINGTON COMPUTING

PROGRAMS LISTED BELOW ARE ON DISK.

Memorex Disks (Special til 4-15-81) **Ten for**		**$34.99**
Apple Galaxian	$24.95 now	**$22.99**
EPYX (Automated Simulations) SPECIAL — While they last: Ryn, Morloc, Rigel all three		**$39.95**
Galaxy Wars	$24.95 now	**$22.99**
Super Script	$89.95 now	**$76.39**
Interactive Fiction	$14.95 now	**$12.99**
Alien Lander	$24.95 now	**$21.99**
Sorcerer: Doom Cavern	$20.00 now	**$17.99**
Hyper Head-On	$24.95 now	**$21.20**
Galactic Empire	$24.95 now	**$21.20**
Galactic Trader	$24.95 now	**$21.20**
Galactic Revolution	$24.95 now	**$21.20**
Dukes Musician	$19.95 now	**$16.95**
Tank Command	$14.95 now	**$12.49**
DB Master	$189.00 now	**$169.00**
Bloody Murder	$19.95 now	**$17.99**
Tranquility Base	$24.95 now	**$22.99**
Aristotle's Apple	$34.99 now	**$29.99**
Micro Memo	$39.95 now	**$35.99**
The Data Factory	$150.00 now	**$134.00**
Mission Asteroid	$19.95 now	**$17.99**
Vinyl holders for 20 disks in beautiful deluxe padded binder		**$12.99**
Skybombers II	$19.95 now	**$17.99**
Apple-Doc	$24.95 now	**$21.99**
Programmers Utility	$16.95 now	**$15.09**
DOS Tool Kit	$75.00 now	**$69.00**
Microsoft Adventure	$29.95 now	**$26.09**
Olympic Decathlon	$29.95 now	**$26.09**
Compu-Math: Arithmetic	$49.95 now	**$44.95**
Apple Plot	$75.00 now	**$69.00**
College Boards (Krell)	$79.95 now	**$69.95**
Time Traveler	$24.95 now	**$22.99**
Sword of Zedek	$24.95 now	**$22.99**
All Time Super Star Baseball	$14.95 now	**$13.19**
CRAE 2.0	$24.95 now	**$21.99**
MCAT 2.0	$19.95 now	**$17.99**
The Tarturian	$24.95 now	**$21.99**
TI PROGRAMMER — Hexadecimal/Octal Calculator Retail $65.00	now	**$59.99**
MPI 88G Printer		**$699.00**
The Prisoner	$29.95 now	**$26.99**
The Wizard & The Princess	$32.95 now	**$28.99**
Compu-Spell	$39.95 now	**$34.99**
Computer Ambush	$59.95 now	**$52.99**
Computer Bismark	$59.95 now	**$52.95**
Computer Napoleonics	$59.95 now	**$52.99**
Computer Quarterback	$39.95 now	**$35.99**
Flight Simulator (disk)	$34.95 now	**$29.30**
Star Cruiser	$24.95 now	**$20.99**
Space Album	$39.95 now	**$34.30**
Odyssey	$29.95 now	**$25.30**
Both Barrels	$24.95 now	**$20.99**
Modifiable Database	$79.50 now	**$66.99**
Micro League Baseball	$24.95 now	**$22.99**
Sargon II	$34.95 now	**$29.70**
Program Line Editor	$40.00 now	**$34.99**
Retro-Ball	$29.95 now	**$25.99**
Space Eggs	$29.95 now	**$25.99**
Mystery House in French		**$22.99**
Z-80 Softcard with CP/M	$349.00 now	**$299.00**
Videx 80-Col. Board	$350.00 now	**$299.00**
EZ Draw	$34.95 now	**$29.99**
Head On	$24.95 now	**$20.99**
3D Super Graphics	$39.95 now	**$35.99**
Compu-Math for II	$40.00 now	**$34.95**
Hi-RES Cribbage	$24.95 now	**$21.99**
Phantom Five	$29.95 now	**$25.49**
Star Gazer's Guide	$30.00 now	**$25.39**
Lords of Karma	$20.00 now	**$16.99**
Tawala's Last Redoubt	$29.95 now	**$25.49**
Apple PIE & Formatter (Reg. $129.95)	Special	**$99.99**
Apple World	$59.95 now	**$52.99**
ABM (Muse)	$24.95 now	**$22.49**
Data Plot (Muse)	$59.95 now	**$52.99**
Computer Conflict	$39.95 now	**$35.99**
Computer Air Combat	$59.95 now	**$52.99**
Terrorist	$29.95 now	**$25.45**
The Temple of Apshai	$39.95 now	**$33.95**
Super-Text II	$150.00 now	**$129.95**
Magic Window	$99.95 now	**$85.00**
CCA Data Management	$99.95 now	**$85.00**
VISICALC	$150.00 now	**$119.00**
Acant: Fortune-telling	$15.00 now	**$8.99**
Hornbloscope	$15.00 now	**$8.99**
Gomoku (cass.)	$14.95 now	**$11.99**
Reversal	$34.95 now	**$29.70**
16K NEC Memory		**$49.99**
Savage Island Adv. (cass.)	$14.95 now	**$12.70**
Baker's Trilogy	$24.95 now	**$21.99**
Crosswords	$29.95 now	**$25.99**
Microgammon II	$19.95 now	**$17.09**
Persoft	$29.95 now	**$25.99**
Dogfight	$29.95 now	**$25.99**
-410G Graphics	$34.95 now	**$29.99**
-460G Graphics	$34.95 now	**$29.99**
Cyber Strike	$39.95 now	**$36.99**
HI-RES Football (On-Line)	$39.95 now	**$36.99**
Fastgammon	$24.95 now	**$21.20**
The Voice	$39.95 now	**$35.99**
Hellfire Warrior	$29.95 now	**$25.45**
Beneath Apple Manor	$20.00 now	**$17.45**
Astro Apple	$20.00 now	**$17.45**
Akalabeth	$34.95 now	**$29.99**
ANDROMEDA 16K RAM Expansion Board for Apple II Retail is $195 — Our price		**$184.99**
NEC 12" Green Black	$260 now	**$219.00**
Centronics 730 Printer	$795 now	**$649.00**
Centronics 737 Printer	$995 now	**$799.00**
ABT Pad for Apple	$125 now	**$109.00**
ROM + w/ filter	$200 now	**$170.00**
M&R 80 col. Video Card	$375 now	**$325.00**
Verbawriter	$249 now	**$219.00**

Send for TRS-80, PET and ATARI Catalog also.

apple SOFTWARE

SPECIAL up to 40% OFF LIST

FASTEST SERVICE

We take MasterCard or VISA (include card # and expiration date). California residents add 6% tax. Include $2.00 for postage. $5.00 Foreign. Mail to:

HUNTINGTON COMPUTING, Dept. COM-3
P. O. Box 787, 2020 Charles, Corcoran, CA 93212
24-hour order service Cal.
(209) 992-5411. Send for free catalog.

```
10  REM  LOAD THE ROUTINE -
    NORMAL GOSUB
20  GOSUB 5000
30  REM  ESTABLISH THE
    AMPERSAND VECTOR
40  CALL 768
50  REM ESTABLISH NAMES OF SUBROUTINES
60 JOHN = 1000
70 PETE = 2000
80 PHREDD = 3000
90  REM  CALL THE SUBROUTINES
100  &  GOSUB JOHN
110  &  GOSUB PETE
120  &  GOTO PHREDD
130  PRINT "DIDNOT WORK"
140  STOP
1000  PRINT "THIS IS JOHN"
1010  PRINT "HERE BY A NAMED GOSUB"
1020  PRINT : RETURN
2000  PRINT "THIS IS PETE"
2010  PRINT "HERE BY A DIFFERENT
      NAMED GOSUB"
2020  PRINT : RETURN
3000  PRINT "THIS IS PHREDD"
3010  PRINT "HERE BY A NAMED GOTO"
3020  STOP
4990  REM  LOAD IN ROUTINE
5000 LOW = 768:HIGH = 835
5010 OK = 1
5020  REM  LOAD IN GROUP OF SIXTEEN
5030  FOR J = LOW TO HIGH STEP 16
5040 CHECK = 0
5050  FOR K = J TO J + 15
5060  READ IT
5070 CHECK = CHECK + IT
5080  NEXT K
5090  REM  CHECK IF CHECK SUM OKAY
5100  READ SUM
5110 L$ = "OKAY": IF CHECK <  > SUM
     THEN L$ = "BAD":OK = 0
5120  PRINT L$
5130  NEXT J
5140  IF OK = 0 THEN  STOP
5150  REM  THINGS ARE OKAY - LOAD INTO MEMORY
5160  RESTORE : FOR J = LOW TO HIGH STEP 16
5170  FOR K = J TO J + 15: READ IT: POKE K,IT: NEXT K
5180  READ IT: NEXT J
5190  PRINT "BLOAD OKAY": PRINT : PRINT
5200  RETURN
6000  DATA  169,76,141,245,3,169,16,141,246
6010  DATA 3,169,3,141,247,3,96,1868
6020  DATA   201,176,240,9,201,171,240,31,162
6030  DATA 16,76,18,212,169,3,32,1957
6040  DATA 214,211,165,185,72,165,184,72,165
6050  DATA 118,72,165,117,72,169,176,2322
6060  DATA 72,32,55,3,76,210,215,32,177
6070  DATA 0,32,123,221,32,82,231,1593
6080  DATA  76,65,217,0,0,0,0,0,0
6090  DATA  0,0,0,0,0,0,0,358
```

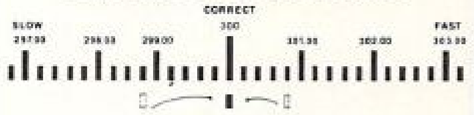
★★★★★★★★★★★★★★★★★★

Bugs in your Apple?

DDT

Disco-Tech's Disc Drive Timer program

zaps disc drive problems!



Analyze disc drive motor speed on a routine basis with an adjustable real-time speedometer. Accurate to one-tenth of one r.p.m. out of 300 r.p.m.

Fine-tune disc drive motor speed yourself. All you need is DDT, two screwdrivers, and five minutes' time.

$29.95 Postpaid
California residents add 6% tax.
Diskette & complete manual.

Engineering Business
Architecture Utilities
Surveying

Also available for TRS-80 Model I.

To order or for more information, write or call

disco tech T.M.

microcomputer products
a division of
Morton Technologies, Inc.
1150 Coddingtown Center
P.O. Box 11129 • Santa Rosa, CA 95406
707/523-1600
Dealer inquiries invited.

★★★★★★★★★★★★★★★★★

```
SOURCE FILE: GOSUB1
------ NEXT OBJECT FILE NAME IS GOSUB1.OBJ0
0300:                        ORG   $300
0300:               ;
0300:               ; M.R.SMITH  APRIL 1981
0300:               ;
0075:               CLINL    EQU   $75         ;CURRENT LINE NUMBER
0076:               CLINH    EQU   $76
00B1:               GETCHR   EQU   $B1         ;GET NEXT CHAR
00B8:               TXTPTL   EQU   $B8         ;TEXT POINTER
00B9:               TXTPTH   EQU   $B9
0300:               ;
D3D6:               STACK    EQU   $D3D6       ;CHECK ON STACK POINTER
D412:               WRONG    EQU   $D412       ;PRINT SYNTAX ERROR MESSAGE
D7D2:               NGOSUB   EQU   $D7D2       ;JUMP INTO NORMAL MONITOR GOSUB
D941:               NGOTO    EQU   $D941       ;JUMP INTO NORMAL MONITOR GOTO
DD7B:               FRMEVL   EQU   $DD7B       ;PUSH VALUE IN FAC
E752:               FIXGOTO  EQU   $E752       ;USE FAC AS GOTO POINTER
0300:               ;
0300:               ;FIX AMPERSAND VECTOR
0300:A9 4C          FIX      LDA   #$4C
0302:8D F5 03                STA   $3F5
0305:A9 10                   LDA   #$10
0307:8D F6 03                STA   $3F6
030A:A9 03                   LDA   #$3
030C:8D F7 03                STA   $3F7
030F:60                      RTS
0310:               ;
0310:C9 B0          ENTRY    CMP   #$B0        ;IS IT GOSUB?
0312:F0 09                   BEQ   GOSUB
0314:C9 AB                   CMP   #$AB        ;IS IT GOTO?
0316:F0 1F                   BEQ   GOTO
0318:A2 10                   LDX   #$10        ;FORCE SYNTAX ERROR MESSAGE
031A:4C 12 D4                JMP   WRONG
031D:               ;
031D:A9 03          GOSUB    LDA   #$3         ;NORMAL GOSUB PROCEDURE
031F:20 D6 D3                JSR   STACK       ;RELOCATED FROM $D921
0322:A5 B9                   LDA   TXTPTH      ;STORE CURRENT TEXT POINTERS
0324:48                      PHA
0325:A5 B8                   LDA   TXTPTL
0327:48                      PHA
0328:A5 76                   LDA   CLINH       ;STORE CURRENT LINE NUMBER
032A:48                      PHA
032B:A5 75                   LDA   CLINL
032D:48                      PHA
032E:A9 B0                   LDA   #$B0        ;IT NEEDS THIS
0330:48                      PHA
0331:20 37 03                JSR   GOTO        ;DO A GOTO
0334:4C D2 D7                JMP   NGOSUB      ;CONTINUE NORMAL GOSUB
0337:               ;
0337:20 B1 00       GOTO     JSR   GETCHR      ;GET NEXT CHAR
033A:20 7B DD                JSR   FRMEVL      ;EVALUATE NEXT EXPRESSION
033D:20 52 E7                JSR   FIXGOTO     ;FIX GOTO LOCATION
0340:4C 41 D9                JMP   NGOTO       ;CONTINUE NORMAL GOTO ROUTINE
```

©

Commas, Colons And Quote Marks Too

Craig Peterson
Santa Monica, CA

Have you ever wanted to be able to input commas, colons or quotation marks as part of an input statement to one of your Applesoft programs? But, hard as you may try, Applesoft kept coming back with "EXTRA IGNORED." Contact 4 from Apple Computer, Inc., helped you by suggesting the use of the GET statement, but all that B$ = B$ + A$ stuff meant that you often had to endure string garbage cleanup delays. Then Contact 6 seemed to offer the ultimate solution, totally avoiding garbage collection. But was it? Besides requiring a small machine language program, there was a subtle problem you might not have been aware of. The input routine used to fill the input buffer made no allowance for the high bit of each character in the input line. The routine used to fill the input buffer left the high bit set, just as it comes from the keyboard. But Applesoft wants the high bit to be zero for its string characters. The line will print correctly and will look on the screen just like what you typed in, but if you ever try an IF IN$ = "Q", you'll never get a match. Or if you try to VAL (IN$),when IN$ was input as "1234", you'll get a value of 0.

The solution to this dilemma is in the program listed below. The subroutine shown in lines 1000 to 1020 (for Applesoft ROM Basic) will gather any input for you and place it into the variable IN$, even commas, colons and quote marks. The only exempt characters are the standard keyboard escape sequences. So, who is the little man at 54572? Well, he's the Applesoft equivalent of the monitor's keyboard input routine, with the difference being that he strips the high bit from all of the input characters. So line 1000 fills the input buffer with normal Applesoft string characters gathered from the keyboard. Line 1010 finds the length of the string, and line 1020 finds the IN$ variable and stuffs its pointers with the right info to point to the keyboard buffer. Then IN$ is relocated into RAM, away from the keyboard buffer. It is not necessary for IN$ to be the first variable used in the program. Lines 1000-1020 can be placed anywhere in your program. The pointers for IN$ are found through the magic of locations 131 and 132, which hold the address of the pointers for the last used variable. It's fast, it totally avoids string garbage build-up, and it's done in Basic. None of that nasty machine language stuff.

One additional note. Not only does this routine work slick for keyboard input, but it also performs the same super feat for disk input, which can be real handy. Commas, etc., in the middle of a name file cause no difficulty when read from the disk. Please note, however, that this routine limits the size of an input string to 239 characters just like the Applesoft INPUT statement does.

So if you need it, try it. It's an easy solution to a common problem.

```
10   HOME : VTAB 4: PRINT "INPUT A
      NYTHING THAT YOU WANT..": PRINT
      : GOSUB 1000: PRINT : PRINT
      "VOILA..": PRINT : PRINT IN$
      : END
20 :
30   REM   LINES 1000 TO 1020 ARE
      A SUBROUTINE THAT PUTS ANY
      INPUT INTO IN$
40 :
1000  CALL 54572
1010  FOR B = 512 TO 751: IF  PEEK
      (B) <  > 0 THEN  NEXT
1020 IN$ = "": POKE  PEEK (131) +
      256 *  PEEK (132) + 1,0: POKE
       PEEK (131) + 256 *  PEEK (1
      32) + 2,2: POKE  PEEK (131) +
      256 *  PEEK (132),B - 512:IN
      $ =  MID$ (IN$,1): RETURN     ©
```

Odds And Ends are growing... Send in your one or two paragraph programming hints to Odds and Ends, c/o **COMPUTE!** P.O. Box 5406, Greensboro, NC 27403 USA

Generating Lower Case Text On The Apple II Plus Using The Paymar Chip

David Shapiro
Bloomington, IN

Introduction

The following program will allow lower case text to be displayed on an Apple II Plus which is equipped with a Paymar chip. The hardware requirements involve the "older" Apple with RAM configuration blocks (an "I.C. impersonator" which only contains jumper wires and is labeled with "16K"), and the PAYMAR lower case adapter, presently advertised as the "original LCA-1 (TM)". By appending this routine to a BASIC program, lower case characters can be embedded inside of quotation marks following a PRINT command by simply converting the corresponding upper case character in the given string. When the BASIC statement involving the PRINT command and the string are executed, the display of upper/lower case text is immediate. Lower case characters can also be converted back to upper case using this routine.

Sample Use Of The Lower Case Converter

Once this routine is appended to a BASIC program, it can then be used for converting between upper and lower case characters:

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz

A typical program statement may contain the string "POUR THE SOLUTIONS." and lower case conversion may be desired on all characters after the "P". The following brief example initially LISTs the statement containing this string, the lower case converter program (which starts at line number 63000) is then RUN, and finally the statement containing the now-converted text is reLISTed.

```
]LIST20
20 PRINT "POUR THE SOLUTIONS."
]RUN63000
WHAT LINE DO YOU WANT CONVERTED? 20
I HAVE FOUND THE LINE.
POUR THE SOLUTIONS.
DO YOU WANT TO CHANGE ANYTHING?
START WITH WHICH CHARACTER? 2
END WITH WHICH CHARACTER? 16
Pour the solutions.
DO YOU WANT TO CHANGE ANYTHING?
]LIST20
20 PRINT "Pour the solutions."
```

The program initially prompts the user for the line number of the BASIC statement to be converted. A search through the Apple's RAM continues until that line number is found, whereupon the characters within quotatin marks are then displayed (if no such line number exists, the program informs the user). A decision to change the string contents is then entered (Y in this case). Character limits for the conversion are individually entered, with only the characters from the upper/lower case sets (see above) sequentially counted (the spaces on either side of "THE" were ignored). The conversion will then start with "O" (the 2nd character) and terminate with the final "S" (the 16th character), with the resultant form displayed for more changes. No further changes were made (input of "N"), and the RESET key was pressed to terminate execution of this routine. This particular statement was then re-LISTed, displaying the quote-embedded lower case text.

More Lower Case Converter Details

The case conversion occurs between the user-defined limits in a continuous fashion. If there are two (or more) separated segments in the same string that are to be converted, then each segment conversion must be done individually. The string is re-displayed after each conversion for further changes if so desired. An individual character can also be converted if the lower and upper numerical limits are identical.

The first time "RUN 63000" is executed, the search for the input line number commences at the beginninng of the program. This search examines the appropriate locations in RAM which the program currently occupies, and with each new examination moves sequentially through the program (increasingly higher memory locations) in an attempt to find the line number. A variable (ML) contains the current RAM location when the line is eventually found. After making the necessary character changes in this statement as stipulated by the user, the search for the next line number will begin at this present memory location (ML). This optimizes the speed with which the program searches for the next line number. If the next line number is less than the last line number, or if it does not exist in the program, then the current RAM location variable ML is re-initialized to zero. The user is informed that the line can not be found, and the next line number search must start at the beginning of the program. This unnecessarily increases the search time; therefore, for maximum speed-execution of the program, all entered line numbers must exist in the program, and they should be entered by increasing value.

The case conversion between upper/lower case in reciprocal; i.e., designated upper case characters will be converted to lower case, and lower case characters will be changed to upper case. Also, if the cursor is used to read a BASIC statement containing a string, any lower case characters will be converted back to upper case (an easy method for converting a mixed-case string to all upper case).

The line numbering of this routine begins at 63000 since lower line numbers should always be used when writing a BASIC program. It may be entered after the END command and accessed at the user's convenience. Typing "RUN 63000" from the keyboard RUNs the routine; pressing the RESET key will terminate its execution.

Program Listing And Explanation

63000 Line number to be converted input as LN.
63010 Initialization of ML to start of BASIC on first RUN of program or when line number is not found; ML is the memory location currently being examined.
63020 NL equated to RAM location of start of next BASIC statement. TL is equated to the line number of BASIC currently being examined.
63030 Jump from search loop if line number is found.
63040 Jump from search loop if line number is not found.
63050 Equate ML to RAM location of the next BASIC statement.
63070 Loop to examine each character/token in the current BASIC statement. Check for quotation mark (ASCII code =34). MODE is a "toggle"; set to 0 when first quote is found.
63080 Printing of characters after 1st quote and up to 2nd quote.
63090 Close PRINT loop.
63100 If no changes ("N") execution transferred to 63000. All other input (including "Y") defaults to 63110.
63110-63120 Limits to define character conversion.
63130 Loop examination of each character/token in BASIC statement. When 1st quote is found, MODE is set to 0.
63140 If the character is between quotes and alphabetic, then counter PO is incremented. When the counter is between the stipulated character limits, the character is converted to upper case (add 32) or lower case (subtract 32) depending on the original value of Q.
63150 Close conversion loop. Control transferred to 63070 for any further changes.

```
63000  INPUT "WHAT LINE DO YOU WA
     NT CONVERTED? ";LN
63010  IF ML = 0 THEN ML = 256 *
      PEEK (104) +  PEEK (103)
63020 NL =  PEEK (ML) + 256 *  PEEK
     (ML + 1):TL =  PEEK (ML + 2)
      + 256 *  PEEK (ML + 3)
63030  IF TL = LN THEN 63060
63040  IF NL < ML OR TL > LN THEN
      PRINT "LINE NOT FOUND.":ML =
     0: PRINT : GOTO 63000
63050 ML = NL: GOTO 63020
63060  PRINT "I HAVE FOUND THE LI
     NE."
63070  PRINT :MODE = 1: FOR A = M
     L + 4 TO NL:Q =  PEEK (A): IF
     Q = 34 THEN MODE = 1 - MODE
63080  IF MODE = 0 AND Q <  > 34 THEN
      PRINT  CHR$ (Q);
63090  NEXT : PRINT
63100  PRINT : PRINT "DO YOU WANT
      TO CHANGE ANYTHING? ";: GET
     A$: PRINT : IF A$ = "N" THEN
     63000
63110  INPUT "START WITH WHICH CH
     ARACTER? ";S
63120  INPUT "END WITH WHICH CHAR
     ACTER? ";E:PO = 0
63130 MODE = 1: FOR A = ML + 4 TO
     NL:Q =  PEEK (A): IF Q = 34 THEN
     MODE = 1 - MODE
63140  IF MODE = 0 AND Q > 64 AND
     Q < 128 THEN PO = PO + 1: IF
     PO >  = S AND PO <  = E THEN
      POKE A,Q + 32: IF Q > 96 THEN
      POKE A,Q - 32
63150  NEXT : GOTO 63070
```
©

Apple Authors

COMPUTE! is looking for applications articles aimed at beginners and intermediate programmers. We're specifically interested in programming hints, tutorials, articles written to help users get more out of their machine.

Apple II High Resolution Character Generator

Peter Gehris and Ken Reinert
Wyomissing, PA

Are you dissatisfied with the small size of the characters on the video screen? Do you have need for a character generator for TV or videotape recording? Would you like to display your own shapes in variable size and position on the screen? Here is a way to achieve these goals using the shape table provision of the APPLE II. The amazing aspects of this program are: (1) you can vary the size of each shape; (2) you can change or add shapes as desired; (3) you can rotate the shapes on the screen even to the point of displaying them upside down or backwards; and (4) you can customize this program to your own programs to yield variable letter forms, sizes and positions for neater looking displays of data and text. The program which positions and draws each character uses high resolution graphics on the video monitor. A copy of the program is in diagram 2. All of the keyboard characters are available except the ›@› which is used as an underline. The instructions to draw each character are stored in a shape table. The creation of the shape table, which is in Chapter 9 of the APPLESOFT II Basic Programming Reference Manual, will not be discussed here.

The Shape Table

In diagram 1, you will see the hexadecimal codes for a shape table beginning at address 37000 (hex 9088) and ending at address 37944 (hex 9440). Please note these addresses are for a 48K system. This table provides the shapes for all characters on the keyboard, except the ›@› key, which is an underline. The shape table should fit right below DOS. HIMEM must then be set at the beginning of the table to protect it from being written over by APPLESOFT variables.

Use this chart to see where the table should be loaded into your system:

Memory	Address				
size	dec.	hex.	HIMEM:	POKE 232,	POKE 233,
32K	21000	$5208	21000	8	82
36K	25000	$61A8	25000	168	97
48K	37000	$9088	37000	136	144

(NOTE: the POKEs will be explained later.)

Loading The Shape Table

By now, the shapes should be defined and converted into hexadecimal code. Now, the codes can be typed into the memory. (It is best if two people do this: one to read the codes and another to type them in.) First, get into the monitor by typing CALL -151. Then, type the address of the shape table: 9088 (for a 48K system) followed by a colon and up to 127 hex codes. The start address will vary according to system size (see chart). The display should look like this:

```
9088: 3B 00 78 00 7A 00 81 00 ...
```

(see page 44 of the APPLE II Reference Manual.) After pressing ‹RETURN›, type the new address (start address plus the number of codes entered) followed by a colon and more hex codes.

To save the shape table, type: 9088,9437W (for tape) or BSAVE (name), A$9088,L$3B0 (for disk). (These addresses are for the shape table listed.) Then, to load it back into memory, type: 9088,9437R (for tape) or BLOAD (name) (for disk), and the table will load into memory, ready for use.

Shape Table Demo Program

The list of the program in diagram 2 will demonstrate the use of the shape table in developing characters on the video screen. Note that for this demo you can control the rotation, scale, position and space between letters through input. We have used a GOSUB in line 60 and a RETURN to show that this can be used as a subroutine within a program. The variables P$, A, B, H, V and Z can be defined by ways other than input prior to each use of the subroutine. The screen is divided into 31 horizontal positions and 17 vertical positions. The rotation angles start at 0 for an upright character. Increasing the rotation equal to 16 will set the shape on its right side (90 degrees); 32 will invert the character 180 degrees, etc. A rotation of 64 will return the shape to an upright position.

A scale of one prints the characters at their normal size (double the size of text). A scale of two doubles the size, a scale of three triples the size, etc. A normal hi-res screen has 280 horizontal dots and

192 vertical dots. Line 1030 converts the H,V coordinates of the 31x17 screen to the X,Y coordinates of the 280x192 hi-res screen. The FOR-NEXT loop at line 1050 prints out the string, one character at a time. Each character is picked out of the string by the MID$ function. That character's ASCII code, minus 31, is used to identify which shape number to draw. Line 1060 increments the X-coordinate to the new position to print the next character in the string.

Line 20 tells the computer where the shape table begins. This is where POKE 232 and POKE 233 are used. Line 40 sets himem to the beginning of the table, so the table is not written over by variables.

Diagram 3 shows another program which uses the secondary page (page 2) of hi-res graphics, which is available only on a 36K or a 48K system. The variables are defined within the program instead of by using input. The routine to enlarge the letters is located at lines 1000-1020.

Customizing The Program

The routine for enlarging the characters can be added to most any program. However, the display cannot be output to a printer, since it is in the hi-res graphics mode. Just remember that there are only 17 lines of 31 characters per screen page (at a scale of 1). With this in mind, you should be up and running with the enlarged characters in no time.

Shapes can be added to the shape table; however only the more advanced programmers should attempt this, as it does require moving blocks of the shape table around and manipulating the shape table index. More on this can be found in Chapter 9 of the APPLESOFT Manual.

Summary

Without a doubt, the work for this program involves the construction of the shape table. Many hours can be spent drawing, plotting vectors, coding in binary and hexadecimal and typing the codes. The program to use the shape table is fairly simple. Uses for the creation of shapes of both usual and unusual types are numerous. Besides, generating the normal alphanumeric and graphics characters, special characters can be created for math/science, foreign languages, etc., and the basic ideas can also be applied to music, art and in creating games.

```
APPLE II HI RESOLUTION GRAPHICS GENERATOR
P. GEHRIS AND K. REINERT   4/8/81


1  REM
2  REM      HI RES DEMO PROGRAM #1
3  REM          FOR 32K SYSTEM
4  REM
10  REM   POKE SHAPE TABLE ADDRESS
20  POKE 232,8: POKE 233,82
30  REM   SET HIMEM TO BEGINNING OF TABLE
40  HIMEM: 21000
50  REM   INPUT DATA
60  TEXT : HOME : INPUT "ENTER STRING : ";P$
70  INPUT "ENTER ROTATION : ";A
80  INPUT "ENTER SCALE : ";B
90  INPUT "ENTER HORIZONTAL POSITION :  ";H
100  INPUT "ENTER VERTICAL POSITION : ";V
110  INPUT "ENTER SPACES BETWEEN LETTERS";Z
120  REM   CALL SUB. TO PRINT STRING
130  GOSUB 1000
140  REM   DELAY, THEN CLEAR SCREEN
150  FOR T = 1 TO 7500: NEXT T: HOME : PRINT  CHR$ (7): GOTO 50
1000  REM    CLEAR SCREEN OF TEXT, PRODUCES FULL PAGE OF HI-RES
      GRAPHICS,SETS COLOR, ROT-ATION AND SCALE
1010  HOME : HGR : POKE 49234,0: HCOLOR= 3: ROT= A: SCALE= B
1020  REM   COMPUTE LOCATION TO PRINT STRING
1030 X = 9 * H - 7:Y =  INT (11 * V - 8):P =  LEN (P$)
1040  REM   PRINT STRING ONE CHARACTER AT A TIME
1050  FOR I = 1 TO P: DRAW  ASC ( MID$ (P$,I,1)) - 31 AT X,Y
1060 X = X + ((9 * B) + Z)
1070  NEXT I
1080  RETURN
```

```
APPLE II HI RESOLUTION GRAPHICS GENERATOR
P. GEHRIS AND K. REINERT   4/8/81

1  REM
2  REM      HI RES DEMO PROGRAM #2
3  REM           FOR 48K SYSTEM
4  REM
10  POKE 232,136: POKE 233,144: REM  POKES SHAPE TABLE STARTING
      ADDRESS
20  HIMEM: 37000: REM   SETS HIMEM TO BEGINNING OF SHAPE TABLE

30  IF  PEEK (37000) <  > 59 THEN  PRINT  CHR$ (4)"BLOAD SHAPES
     ": REM  LOADS SHAPE TABLE INTO MEMORY IF NOT ALREADY IN ME
     MORY
40  HOME : HGR2 : POKE 49234,0: HCOLOR= 3: ROT= 0: SCALE= 1: REM
       SETS HI-RES GRAPHICS PARAMETERS; USES PAGE 2 OF HI-RES M
     EMORY
50 P$ = "THIS DEMONSTRATION PROGRAM":H = 4:V = 2: GOSUB 1000
60 P$ = "USES THE HI-RES":H = 9:V = 5: GOSUB 1000
70 P$ = "CHARACTER GENERATOR":H = 7:V = 8: GOSUB 1000
80 P$ = "AND SHAPE TABLE":H = 9:V = 11: GOSUB 1000
90 P$ = "TO PRINT THIS":H = 10:V = 14: GOSUB 1000
100 P$ = "DISPLAY":H = 13:V = 17: GOSUB 1000
110  FOR W = 1 TO 5000: NEXT
120  TEXT : HOME : END
1000 X = 9 * H - 7:Y =  INT (11 * V - 8):P =  LEN (P$)
1010  FOR I = 1 TO P: DRAW  ASC ( MID$ (P$,I,1)) - 31 AT X,Y:X =
     X + 9
1020  FOR T = 1 TO 50: NEXT T: NEXT I: RETURN
```

```
9088- 3B 00 78 00 7A 00 81 00
9090- 88 00 99 00 AD 00 C3 00
9098- D6 00 DA 00 E5 00 EE 00
90A0- 01 01 0C 01 14 01 1A 01
90A8- 21 01 2B 01 3C 01 48 01
90B0- 59 01 6A 01 7B 01 8B 01
90B8- 9F 01 AB 01 BE 01 D3 01
90C0- DA 01 E3 01 EF 01 F9 01
90C8- 03 02 0E 02 15 02 27 02
90D0- 3B 02 4A 02 5A 02 6C 02
90D8- 7B 02 8D 02 9E 02 AA 02
90E0- B7 02 CB 02 D1 02 E3 02
90E8- F6 02 06 03 16 03 28 03
90F0- 3C 03 4D 03 57 03 65 03
90F8- 73 03 85 03 95 03 A1 03
9100- 01 00 49 36 36 36 96 06
9108- 00 31 36 4D 21 24 04 00
9110- 89 36 36 36 6E 21 24 24
9118- 24 95 1F FF 96 0D 6D 05
9120- 00 49 36 36 36 36 26 D8
9128- AB 6D 2D 20 E4 FF 3F 20
9130- 0C 6D AD 05 00 32 0E 05
9138- 20 07 28 2D 2D 36 1E 1E
9140- 1E 1E 1E 1E 4D 21 0C 15
9148- F6 05 00 12 76 0E 0E 0E
9150- 0D 16 1C 1E 3F 3F 20 0C
9158- 0C 08 24 1C 3F 00 49 36
9160- 36 00 49 09 1E 1E 1E 36
9168- 76 0E 0E 06 00 A9 15 15
```

```
9170- 36 F6 1E 1E 06 00 72 0E
9178- 8E 71 0E DE 23 24 24 24
9180- 6C 11 17 17 D7 BA 17 07
9188- 00 92 2A 2D 2D B5 DA 23
9190- 24 20 24 00 49 92 92 37
9198- 35 1E 06 00 92 52 2D 2D
91A0- 5A 00 92 92 52 29 3E 04
91A8- 00 49 49 F2 1E 1E 1E 1E
91B0- 1E 06 00 12 36 36 76 0E
91B8- 2D 05 28 20 24 24 1C 1C
91C0- 3F 17 05 00 2D 35 36 36
91C8- 36 36 3F 6F 09 2D 05 00
91D0- 12 0C 0C 2D 2D 32 1E 1E
91D8- 1E 1E 1E 1E 2E 2D 2D 2D
91E0- 00 2A 28 2D AD 36 1E 3F
91E8- 37 49 31 36 1E 3F 3F 07
91F0- 20 00 49 29 36 36 2E 96
91F8- 1A 24 24 3F 3F 27 05 28
9200- 28 28 00 2D 2D 2D DE DB
9208- 33 36 2D 2D AD 36 F6 3F
9210- 3F 27 00 92 32 36 76 2D
9218- 2D 05 20 24 1C 3F 3F 07
9220- 20 64 2D 2D 15 05 00 2D
9228- 2D 2D 36 1E 1E 1E 1E 1E
9230- 36 05 00 32 B6 36 76 2D
9238- 2D 05 20 24 04 20 E4 3F
9240- 3F 96 2A 2D 2D 00 02 36
9248- 96 12 0E 2D 2D 05 20 24
9250- 24 24 1C 3F 3F 16 12 2D
```

```
9258- 2D 05 00 52 29 3E 96 35
9260- 37 00 52 29 3E 96 35 77
9268- 1E 06 00 49 09 1E 1E 1E
9270- 1E 0E 0E 0E 0E 06 00 12
9278- 2D 2D 2D 96 3A 3F 3F 3F
9280- 00 A9 15 15 15 1E 1E 1E
9288- 1E 06 00 2A 28 2D AD 36
9290- 1E BF 36 16 05 00 93 2D
9298- 2D 2D 2D 05 00 32 36 36
92A0- 36 6E 49 21 24 24 24 E4
92A8- 3F 3F 96 2A 2D 2D 00 36
92B0- 36 36 36 2E 2D 2D 05 20
92B8- 24 04 20 E4 3F 3F 96 2A
92C0- 2D 2D 00 29 2D AD B6 92
92C8- F6 3F 3F 07 20 24 24 24
92D0- 04 00 2D 2D 15 15 36 36
92D8- F6 1E 3F 3F 24 24 24 24
92E0- 24 00 2D 2D 2D DE DB 33
92E8- 36 2D 2D DE 1B 36 36 2D
92F0- 2D 2D 05 00 2D 2D 2D DE
92F8- DB 33 36 2D 2D DE 1B 36
9300- 36 05 00 29 2D AD DF 92
9308- 2A AD 36 1E 3F 3F 07 20
9310- 24 24 24 04 00 36 36 36
9318- 36 6E 49 21 24 24 3F 3F
9320- 67 49 21 24 04 00 29 2D
9328- F5 33 36 36 36 06 3F 4D
9330- 2D 00 49 49 36 36 36 36
9338- 1E 3F 3F 07 20 04 00 36
9340- 36 36 36 6E 49 E1 1C 1C
9348- 1C 1C 0C 0C 0C 0C 05 00
9350- 36 36 36 36 2E 2D 2D 2D
9358- 00 36 36 36 36 6E 49 21
9360- 24 24 24 24 17 17 17 07
9368- 38 28 00 36 36 36 36 6E
9370- 49 21 24 24 24 24 DF 9B
9378- 15 15 15 15 05 00 29 2D
9380- AD 36 36 36 F6 3F 3F 07
9388- 20 24 24 24 04 00 2D 2D
9390- AD 36 1E 3F 3F 04 C0 30
9398- 36 36 36 36 05 00 32 36
93A0- 36 36 0E 2D 6D 1C 07 68
93A8- 24 24 24 1C 3F 3F 07 00
93B0- 36 36 36 36 6E 49 E1 1C
93B8- 1C 1C 07 28 2D 2D 20 E4
93C0- 3F 3F 07 00 29 2D AD DF
93C8- DB 36 0E 2D 2D 15 36 F6
93D0- 3F 3F 07 28 00 2D 2D 2D
93D8- DE 33 36 36 36 36 00 36
93E0- 36 36 36 0E 2D 2D 05 20
93E8- 24 24 24 24 00 36 36 36
93F0- 0E 0E 0E 05 28 28 20 24
93F8- 24 24 00 36 36 36 36 0E
9400- 2D 20 24 24 24 6C 31 36
9408- 36 36 F6 3F 00 76 0E 96
9410- 1E 1E 6E 49 21 1C 1C 1C
9418- 64 0C 0C 24 00 76 0E 16
9420- 92 0A 24 24 64 0C 0C 24
9428- 00 2D 2D 2D 36 1E 1E 1E
9430- 1E 1E 1E 2E 2D 2D 2D 00
```

©

HUNTINGTON COMPUTING

PROGRAMS LISTED BELOW ARE ON DISK:

Memorex Disks (Special 4/4-15-81) **Ten for**		**$24.99**
Apple Galaxian	$24.95 now	**$22.99**
EPYX (Automated Simulations) SPECIAL — While they last: Ryn, Morloc, Rigel all three		**$29.95**
Galaxy Wars	$24.95 now	**$22.99**
Super Script	$89.95 now	**$76.39**
Interactive Fiction	$14.95 now	**$12.99**
Alien Lander	$24.95 now	**$21.99**
Sorcerer: Doom Cavern	$20.00 now	**$17.99**
Hyper Head-On	$24.95 now	**$21.20**
Galactic Empire	$24.95 now	**$21.20**
Galactic Trader	$24.95 now	**$21.20**
Galactic Revolution	$24.95 now	**$21.20**
Golden Mountain	$19.95 now	**$16.95**
Tank Command	$14.95 now	**$12.49**
DB Master	$189.00 now	**$169.00**
Bloody Murder	$19.95 now	**$17.99**
Tranquility Base	$24.95 now	**$22.99**
Aristotle's Apple	$34.95 now	**$29.99**
Micro Memo	$39.95 now	**$33.99**
The Data Factory	$150.00 now	**$134.00**
Mission Asteroid	$19.95 now	**$17.99**
Vinyl holders for 20 disks in beautiful deluxe padded binder		**$13.99**
Skybombers II	$19.95 now	**$17.99**
Apple Doc	$24.95 now	**$21.99**
Programmers Utility	$16.95 now	**$15.09**
DOS Tool Kit	$75.00 now	**$69.00**
Microsoft Adventure	$29.95 now	**$26.09**
Olympic Decathlon	$29.95 now	**$26.09**
Compu-Math: Arithmetic	$49.95 now	**$44.95**
Apple Plot	$75.00 now	**$69.00**
College Boards (Krell)	$79.95 now	**$69.95**
Time Traveler	$24.95 now	**$22.99**
Sword of Zedek	$24.95 now	**$22.99**
All Time Super Star Baseball	$14.95 now	**$13.19**
CRAE 2.0	$24.95 now	**$21.99**
MCAT 2.0	$19.95 now	**$17.99**
The Tarturian	$24.95 now	**$21.99**
TI PROGRAMMER — Hexidecimal/Octal Calculator Retail $65.00	now	**$59.99**
MPI 88G Printer		**$699.00**
The Prisoner	$29.95 now	**$26.99**
The Wizard & The Princess	$32.95 now	**$28.99**
Compu-Spell	$39.95 now	**$34.99**
Computer Ambush	$59.95 now	**$52.99**
Computer Bismark	$59.95 now	**$52.95**
Computer Napoleonics	$59.95 now	**$52.99**
Computer Quarterback	$39.95 now	**$35.99**
Flight Simulator (disk)	$34.95 now	**$29.50**
Star Cruiser	$24.95 now	**$20.99**
Space Album	$39.95 now	**$34.30**
Odyssey	$29.95 now	**$25.30**
Both Barrels	$24.95 now	**$20.99**
Modifiable Database	$79.50 now	**$64.99**
Micro League Baseball	$24.95 now	**$22.99**
Sargon II	$34.95 now	**$29.70**
Program Line Editor	$40.00 now	**$34.99**
Retro-Ball	$29.95 now	**$25.99**
Space Eggs	$29.95 now	**$25.99**
Mystery House in French		**$22.99**
Z-80 Softcard with CP/M	$349.00 now	**$299.00**
Videx 80-Col. Board	$350.00 now	**$299.00**
EZ Draw	$34.95 now	**$29.99**
Head On	$24.95 now	**$20.99**
3D Super Graphics	$39.95 now	**$35.99**
Compu-Math For II	$40.00 now	**$34** 00
HI-RES Cribbage	$24.95 now	**$21.99**
Phantom Five	$29.95 now	**$25.49**
Star Gazer's Guide	$30.00 now	**$25.39**
Lords of Karma	$20.00 now	**$16.99**
Tawala's Last Redoubt	$29.95 now	**$25.49**
Apple PIE & Formatter (Reg. $129.95)	Special	**$99.99**
Apple World	$59.95 now	**$52.99**
ABM (Muse)	$24.95 now	**$22.49**
Data Plot (Muse)	$59.95 now	**$52.99**
Computer Conflict	$39.95 now	**$35.99**
Computer Air Combat	$59.95 now	**$52.99**
Terrorist	$29.95 now	**$25.45**
The Temple of Apshai	$39.95 now	**$33.95**
Super-Text II	$150.00 now	**$129.95**
Magic Window	$99.95 now	**$85.00**
CCA Data Management	$99.95 now	**$85.00**
VISICALC	$150.00 now	**$119.00**
Acent: Fortune-telling	$15.00 now	**$8.99**
Horriblescope	$15.00 now	**$8.99**
Gomoku (cass.)	$14.95 now	**$11.99**
Reversal	$34.95 now	**$29.70**
16K NEC Memory		**$49.99**
Savage Island Adv. (cass.)	$14.95 now	**$12.70**
Baker's Trilogy	$24.95 now	**$21.99**
Crosswords	$29.95 now	**$25.99**
Microgammon II	$19.95 now	**$17.99**
Pensoft	$29.95 now	**$25.99**
Dogfight	$29.95 now	**$25.99**
440G Graphics	$34.95 now	**$29.99**
460G Graphics	$34.95 now	**$29.99**
Cyber Strike	$39.95 now	**$36.99**
HI-RES Football (On-Line)	$39.95 now	**$36.99**
Fastgammon	$24.95 now	**$21.20**
The Voice	$39.95 now	**$35.99**
Hellfire Warrior	$29.95 now	**$25.45**
Beneath Apple Manor	$20.00 now	**$17.45**
Astro Apple	$20.00 now	**$17.45**
Akalabeth	$34.95 now	**$29.99**
ANDROMEDA 16K RAM Expansion Board for Apple II Retail is $195 — Our price		**$184.99**
NEC 12" Green Black	$260 now	**$219.00**
Centronics 730 Printer	$795 now	**$649.00**
Centronics 737 Printer	$995 now	**$799.00**
ABT Pad for Apple	$125 now	**$109.00**
ROM + w/ filter	$200 now	**$170.00**
M&R 80-col. Video Card	$375 now	**$325.00**
Versawriter	$249 now	**$219.00**

Send for TRS-80, PET and ATARI Catalog also.

apple SOFTWARE

SPECIAL up to 40% OFF LIST

FASTEST SERVICE

We take MasterCard or VISA (include card # and expiration date). California residents add 6% tax. Include $2.00 for postage. $5.00 Foreign. Mail to:

HUNTINGTON COMPUTING, Dept. COM-3
P. O. Box 787, 2020 Charles, Corcoran, CA 93212
24 hour order service. Call
(209) 992-5411. Send for free catalog.

VISA

How Do I Fit A 16K Program Into A 6K Space?

(Simple – You Don't)

J. F. Johnson
University of Notre Dame
Dept. of Chemistry
Notre Dame, IN

When an Apple II Plus is turned on, default values are assigned to a region in random access memory which is used to control certain system functions. Examples of these functions include system machine language programs, "stacks", input lines by programs, and certain visual display modes of the Apple. The first 2048 bytes (or pages 0 through 7, each page comprising 256 bytes) are reserved for these functions, with the balance of RAM utilized for BASIC, machine language programs, binary data, or DOS. Figure 1 depicts the outlay of user-accessible memory assuming a 48K Apple, with both HI-RES pages also included.

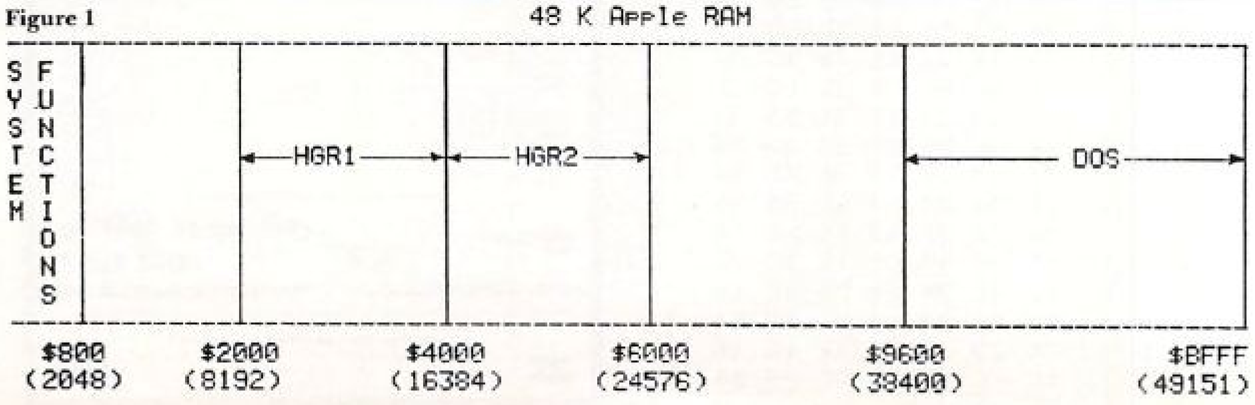
Other areas of RAM may also be used by peripheral devices or functions that are included in Applesoft (the version of BASIC designed for the Apple). If the Apple is interfaced to a disk drive, the disk operating system (DOS) is loaded automatically into the top portion of RAM (38400K-49151K on a 48K Apple; a corresponding upper RAM region is used on Apples with less memory). This upper 10K region can be used by a BASIC program if your Apple is cassette supported. The contiguous portion of memory from 8192K-24576K however can generate frustrating problems. If your program exceeds 6K and extends into the first or second HI-RES page, this problem becomes self-evident when the program attempts to use a "BASIC-loaded" HI-RES page.

For example, the Apple by default starts loading a BASIC program at $800 (2048k). And of course one of the excellent features of the Apple is the use of the HI-RES pages for animation or detailed graphics effects. However, when either HI-RES page is used from BASIC (by the use of either the HGR or HGR2 command), the respective HI-RES region in memory is literally filled with zeroes. If your program is larger than 6K bytes (that region from 2K-8K), then the portion of BASIC extending into the first HI-RES page memory region will be destroyed upon use of the HGR command since the 8K-16K region will be "erased". An analogous situation occurs if the program extends into the 16K-24K region of RAM and the HGR2 command is used. A solution to this problem involves loading the BASIC program into a different region of RAM, and hence overriding certain default values that the Apple assigns when it is turned on.

There are two memory locations in the zero page of Apple's RAM which dictate where BASIC programs are loaded (See Applesoft Memory Map (Page 0) by Jim Butterfield, Issue 6 of **COMPUTE!** or page 140 in the Applesoft BASIC Programming Reference Manual). The pointer to the start of a BASIC program is comprised of the most significant byte 104 ($68) and the least significant byte 103 ($67), and their contents may be changed by using the POKE command. By POKEing 104 and 103 prior to loading a program, the portion of RAM used for BASIC storage can be altered.

Most of my programs occupy about 16K of RAM, require the first HI-RES page for graphics, and operate on a 32K diskless Apple II Plus. It should now be recognized that if the pointer to the beginning of the program (locations 104 and 103) is left at the default value of 2048 ($800), approximately the last 10K of my program would be destroyed upon usage of the first HI-RES page. The simplest solution is to load the BASIC program immediately above this HI-RES page in memory, which can be accommplished on either a cassette or disk-supported machine, but must be done prior to loading the program.

Figure 1



For cassette supported systems, after turning on your Apple type the following, then press return.

```
POKE 104,64:POKE 103,1:POKE 16384,0 ‹RET›
```

LOADing of a BASIC program now starts at decimal location 16385 (= 256*64 + 1; the most significant byte is multiplied by 256 and added to the least significant byte for most pointers), and can occupy the upper 16K of RAM (16385K-32768K). The byte immediately before the memory location where LOADing is initiated must be a zero, hence location 16384 contains a zero. (To change this pointer to any RAM location divide the chosen decimal value by 256, POKE the integer portion of the answer into 104 and POKE the remainder into 103.)

The identical results can be obtained on a 48K disk supported Apple, using a slightly more sophisticated method. An EXEC file may be created, which when EXECed by the HELLO program initializes the pointer to the beginning of the program and then loads the program. The following program creates the EXEC file "LOADER".

```
10 D$ = CHR$(4)
20 PRINT D$"OPEN LOADER"
30 PRINT D$"WRITE LOADER"
40 PRINT "POKE 104,64"
50 PRINT "POKE 103,1"
60 PRINT "POKE 16384,0"
70 PRINT "LOAD TITRATION"
80 PRINT "CLOSE LOADER"
```

The following HELLO program would then EXEC the file LOADER.

```
10 D$ = CHR$(4)
20 PRINT D$"EXEC LOADER"
```

The same diskette must of course contain the HELLO program (which is run when the system is booted), the BASIC program "TITRATION", and the EXEC file "LOADER". The use of an EXEC file saves the tedium of POKEing locations from the keyboard, followed by a "LOAD TITRATION" DOS command (which gives the same results but is considered less time-efficient). This now BASIC-vacated region from 2K-8K usually stores shapes which are displayed on the first HI-RES page under control of the newly located BASIC program. ©

Apple Authors

COMPUTE! is looking for applications articles aimed at beginners and intermediate programmers. We're specifically interested in programming hints, tutorials, articles written to help users get more out of their machine.

Editor's Note: Here's the first in a series of assembler programs to enhance your Integer BASIC. RCL

Ever Expanding Apple Power

Mitchell Bushin
Scarsdale, NY

I am an Apple II owner who found Apple Integer Basic a rather limited language. In order to improve the language's power, I have written assembler programs to "attach" to Integer Basic programs.

The first of my sample programs is an idea stolen from Wang Labs. They have a row of Special Function keys on their machine. These keys can be used to allow a user to type a whole word in answer to an input statement by touching one key. I liked the feature and wrote a program that allows an Apple-ite (Apple-user) to use the number keys as special function keys.

This is the Basic part:

```
1   w=0: rem this must be the first statement
100 rem this is a piece of a home accounting program
110 respond=2048: rem address of routine
500 call -936: rem clear screen
510 vtab 10:print "commment on expense:"
520 call respond: if w#1 or w#2 or w#3 then goto 600
530 dim ans$(30): rem the answer we want
540 if w#1 then goto 550:ans$="oil":s=24
550 if w#2 then goto 560:ans$="petty-cash refill":
    s=36
560 if w#3 then goto 570:ans$="electric":s=29
570 vtab 10:tab 20:print ans$: rem automatic ans is
    printed
580 vtab 10:tab s:input a$: rem wait for return to
    continue
590 goto 610
600 vtab 10:tab 20:input ans$
610 rem rest of program
```

Together with the assembler program, this program types an answer automatically when a pseudo-S.F. key is hit. There can be up to 9 different automatic answers. This is good for an office Apple that runs a program that would normally have a specific number typed to give an answer. In addition, with the S.F. keys, hitting the space bar allows the Apple-ite to type in any answer he wants.

The assembler part:

```
loc in hex
800         JSR     $FF4A      SAVE REG.
803         BIT     $C000      IS KEY PRESSED
            BPL     $0803      NO-GOTO 803
            LDA     $C000      WHAT IS CODE
            CMP     #$BA       IS IT A NUMBER
            BCS     $0813      NO-GOTO 813
```

```
        AND     #$0F      WHAT IS THE
                          NUMBER
        BPL     $0815     GOTO 815
813     LDA     #$00      LET W BE ZERO
815     LDY     #$04      W IS THE 4TH BYTE
                          IN VARIABLE LIST
        STA     (4A),Y    4A IS ADDR. OR VAR.
                          LIST
        BIT     $C010     RESET KEY STROBE
        JMP     $FF3F     RETURN REGISTERS
81F
```

After loading the assembler program set lomem so that it does not interfere with the variable list. Lomem:2079 is good enough. This program is portable and can be placed anywhere in memory. ©

Dysan CORPORATION

Solve your disc problems, buy 100% surface tested Dysan diskettes. All orders shipped from stock, within 24 hours. Call toll FREE (800) 235-4137 for prices and information. Visa and Master Card accepted. All orders sent postage paid.

PACIFIC EXCHANGES
100 Foothill Blvd.
San Luis Obispo, CA
93401 (In Cal. call
(805) 543-1037)

Computer House Division

PROGRAMS FOR COMMODORE AND APPLE	
Legal accounting Demo	$15.00
Legal accounting Program	995.00
Machine Part Quote Demo	15.00
Machine Part Quote Program	325.00
Mailing/phone list	80.00
Political Mail/phone list	130.00
Beams, structural	115.00
Trig/Circle Tangent	110.00
Spur Gears	35.00
Bolt Circles	25.00
Filament Wound TAnks	125.00
Scrunch	25.00

PROGRAMS FOR COMMODORE ONLY	
A/P, A/R, Job Cost & Job Est.	370.00
Inventory	95.00
Financial	175.00
Real Estate Listings	265.00
Check Writer	25.00
File Editing Tools (FET)	65.00
Screen Dump/Repeat	35.00
Docu-Print	20.00
Scrunch	25.00
Sof-Bkup	40.00
Sorter (Mach. Language)	35.00
Trace-Print	25.00
Vari-Print	25.00

ASK FOR CATALOG #80-C2 Dealers Wanted
Computer House Div. 1407 Clinton Road
Jackson, Michigan 49202 (517) 782-2132

APPLE II*
SOFTWARE FROM
POWERSOFT
P. O. BOX 157
PITMAN, NEW JERSEY 08071
(609) 589-5500

SUPER CHECKBOOK III—A **vastly improved** version of our popular selling program. With new features such as: simplified but powerful transaction entry and modification routines, new reconciliation routines, additional features such as 30 percent increase in the total number of checks handled, posting of interest from interest bearing checking accounts, automatic teller transactions, bullet proof error handling, and smart disk routines. Plus the program still contains the options of bar graphs, sorting, activities, and account status.
Disk Only/Applesoft $49.95

ADDRESS FILE GENERATOR—Allows the user to create any number of four types of address files: Holiday, Birthday, Home and Commercial. The program contains a menu of seven major commands used to: Create, Add, Edit, Display, Search, Sort, and Reorganize Files. Up to three fields may be used for the sort criteria. Major commands have subordinate commands which adds to the flexibility of this powerful software system. We doubt you could buy a better program for maintaining and printing address files .... Disk Only/ Apple Printer Card/Applesoft $24.95

SPANISH VOCABULARY DRILL
FRENCH VOCABULARY DRILL
ITALIAN VOCABULARY DRILL
GERMAN VOCABULARY DRILL

These programs provide practice in foreign language vocabulary by means of three types of drills: **Matching, Foreign Language to English and English to Foreign Language.** Although the diskette comes with some lessons on it, these are intended to be samples. The most effective way to use these programs is to enter your own lessons from the course you are studying. To facilitate the entry of new lessons, each program contains a complete Lesson Editor which has various entry and revision options. The manual also contains instructions for converting the programs to other languages .... Disk Only/ Applesoft each $24.95

SPACE TREK J—Your mission is to patrol the galaxy, and to seek out and destroy the ships of the Klarian fleet. At your command is the starship Lexington. The Lexington has a wide variety of weapons, sensors and other devices useful in your mission. There are two kinds of Klarian ships Regular and Super. Regular Klarians have an average energy supply of 5000 to 12000 quarks while Super Klarians have 12500 to 15000 quarks and are protected from some of the Lexingtons' weapons and attack strategies .... Disk Only/ Applesoft $19.95

WORLD OF ODYSSEY—An adventure game utilizing the full power of Disk II, which enables the player to explore 353 rooms on 6 different levels full of dragons, dwarfs, orcs, goblins, gold and jewels. The program allows the player to stop the game and to resume at a later time .... Disk Only/Applesoft $24.95

GALACTIC EMPIRES—Pits 1 to 20 players against each other and the computer in a struggle for control of up to 40 star systems. The players compete by sending out fleets of ships to capture neutral planets and to attack the colonies of other players. The victor is the player who controls the most stars by game's end.
Applesoft $14.95

Dealer Inquiries Invited

Visa and MasterCard, Check or Money Order include $2.00 for shipping and handling. C.O.D. $2.00 additional.

*Apple II and Applesoft are registered trademarks of APPLE COMPUTER INC.

POWERSOFT
P. O. BOX 157
PITMAN, NEW JERSEY 08071
(609) 589-5500

Animating Integer BASIC Low-Resolution Graphics

Leslie M. Grimm
Mt. View, CA

Animated low-resolution graphics can add a lot of pizazz to your Basic program. It takes longer to design the program, and requires the knowledge of a few peeks and pokes, but is not really too difficult to learn, and the results make the effort extremely worthwhile.

There are two basic techniques involved. The first is the design of the animated figures. This is similar to what is done in designing cartoon figures. The second technique involves "flipping pages" on the computer while successively drawing each section of your animated figure to create the illusion of movement. Each technique will be explained here, with a short Integer Basic program using the technique as an example.

Designing The Figure

Before you take pencil and paper in hand it is a good idea to spend some time observing or visualizing the object you wish to animate in the various phases of its motion. For the program example here, the figure to be animated was a girl, who was to be shown walking from left to right across the screen. Observation of people walking was followed by paper and pencil sketches of the various stages of the walking motion, as shown in fig. 1. In this example, the illusion of walking can be created with a succession of four pictures, A, B, C, and D, as shown.

Once you have a sketch of your figure, you'll need some graph paper. Quadrille-ruled paper is fine, but the special Apple graphics paper available in some computer outlets is more accurate, because the "squares" on the screen are really rectangles, which can throw off the proportions of your finished figure.

Each phase of the movement of the figure should be drawn on its own rectangular section of the graph paper. Each rectangular section should be the same size, and the figure should be centered in exactly the same position in each rectangle. (See fig. 2). In the case of the girl, the rectangle had to be wide enough to accomodate the figure, whether the leg was projected forwards or backward. An extra space was also allowed on either side of the girl, but this is not essential. Each rectangle should be numbered as shown in Fig. 2, with the upper left-hand corner counted as 0,0 (just as the upper left corner of the graphics screen is 0,0).

Now you are ready to develop the subroutines that will draw the figure. You will need one subroutine for each phase of the movement, as a minimum. These subroutines should be assigned

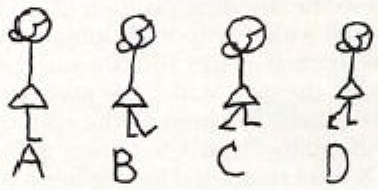


Figure 1

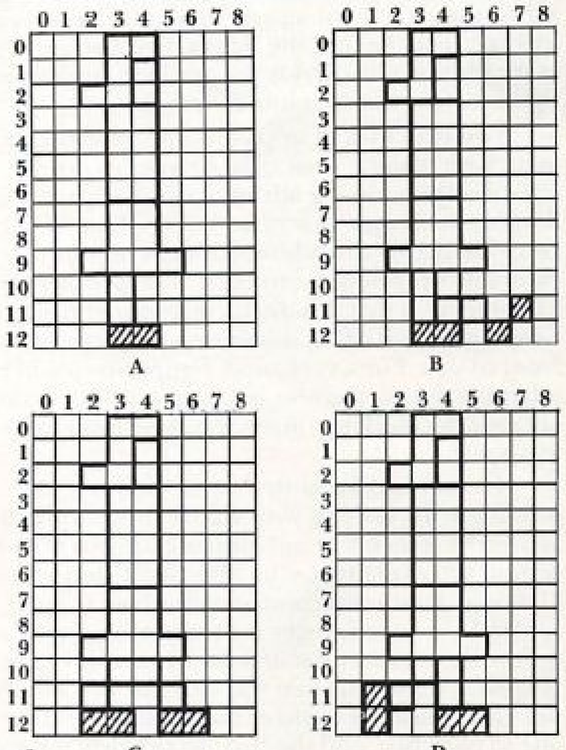


Figure 2

low line numbers, and as many commands as possible should be crammed into each line in order to speed up the drawing time. (The larger your picture is, the more critical this aspect becomes.) Use HLIN and VLIN commands wherever possible.

Since your figure is going to be moving on the screen, your subroutines should not plot specific points (such as PLOT 12,21), but should be written instead in general terms. To do this call the 0,0 square of each rectangle X,Y. Then write your commands in terms of X and Y. For example, the girl's face is drawn with a short vertical line at column 4 in the picture. The *specific* command to draw the face would be VLIN 1,2 at 4. The *general* command is VLIN Y+1,Y+2 at X+4. If X and Y are set to zero before the subroutine is called, the face will be drawn from Y=1 to Y=2 at X=4. It can be moved one space to the right by adding one to X before calling the subroutine.

The subroutines that draw the girl in this program are located from 100 to 200. Subroutine 100 draws the upper half of the girl, which is the same in each picture. Subroutine 120 draws the lower half in the standing position (A). 130 draws the lower half with one foot stepping forward (B), 140 is mid-stride (C), and 150 finishes the sequence (D). To make the girl "walk," the main program calls 100 (top half) followed by the appropriate bottom half, going from A to D over and over. The value for X is incremented by one before each successive step is drawn, so that the girl moves across the screen one square at a time. To prevent leaving a trail behind the figure, you also need an erase routine (160 in this program) to remove each figure before the next one is drawn.

You may wish to stop at this point and experiment with making your figure move across the screen without taking advantage of the page-flipping technique described below. This will give you a chance to add additional drawings or to eliminate drawings not needed. The problem you will observe is that it is distracting to see the figure blinking on and off as it is erased and redrawn in front of you. For a very small figure this could be tolerated, but as pictures get larger or more complicated the blinking causes the animation to lose its appeal.

The remainder of this article describes a technique for making your figure move smoothly across the screen without blinking on and off. One section will explain how to "flip pages" to prevent blinking. Another section explains how to reset LOMEM in your Integer Basic program in order to free the memory needed for the second page of graphics. A third section will describe the subroutine that is used to transfer the contents of page one to page two, and the final section will tell you how the main program works.

Flipping Pages

The Apple has two blocks of memory that can be used for low-resolution graphics. They are referred to as pages. Page one is the one you normally use when you enter the command GR. This page is also used for your text statements. Gaining access to page two requires a bit of trickery. One of the tricks is to prevent this page from being used to store the variables in your Integer program. This is performed by putting the statement LOMEM: 3072 as the first line of the program. Unfortunately, you can't just type that in. (Try it and see for yourself!) But there is a way to do it, which is explained below. The second trick is to put a picture on page two. But alas! there is no command to draw or print text on page two. It can be done, however, by making the drawing on page one, in the usual way, and calling a special subroutine in the Apple Monitor to make a copy of page one on page two. This is also explained below.

Setting LOMEM:

There is more than one way to accomplish this task, but the method described here results in the simplest program. It requires doing some things in the Apple Monitor, but each step will be carefully explained, so you should have no trouble. You may want to save the program you have typed so far (if any) before you begin. If you haven't started a program yet, this will be your first line.

Enter the following as the first line of your Integer program: 0 PRINT 3072. Now type CALL-151 to get into the Monitor. (Fig. 3 shows what you will see as we go along.) You should see the * prompt. The first task is to locate the machine language version of the Basic program line 0 which you just typed in. This is done by looking at the numbers stored in two special memory locations in the Monitor – memory locations 00CA and 00CB (or CA and CB for short.) These locations are called pointers, and contain numbers representing the address of the beginning of your program. (Note: The letters A, B, C, D, E, and F are numbers (10 through 15) in machine language. The 0's in machine language are all zeros, so when you see "0" in this part of the article, type a zero on your computer, not a letter 0.)

Type CB (return). You will see 00CB- ##. The actual number represented here by ## will depend on the size of your program and the amount of memory in your computer. In Fig. 3 the number is 90. Now type CA (return). You will see 00CA– ## (## = C3 in fig. 3). The two-digit numbers you just found are the two halves of the four digit address of line zero of your Basic program. The first half of the four digit number is the one you found at CB, and the second half is the one you found at CA. In the example in fig. 3, the whole four digit number is 90C3. By typing this number (using the actual numbers you found on

```
>CALL-151

*CB

00CB- 90
*CA

00CA- C3
*90C3

90C3- 08
*
 00 00 62 B3
*

90C8- 00 0C 01 1C 0A 00 64 36
*90C3:08 00 00 11

*
```

your computer) and pressing (return) two or three times you will see a series of numbers representing line zero of your Basic program.

The first eight two-digit numbers you should see are the following: 08 00 00 62 B3 00 0C 01. The number "08" means there are eight numbers for line zero; the numbers "00 00" mean the first line is line zero; the "62" means PRINT; "B3 00 0C" stands for 3072 and the "01" at the end means "end of this line." By replacing the number 62 with the number 11 you can change the PRINT in line zero to LOMEN:.

To change the 62 to 11, just type in the address you found in CB and CA again (90C3 in fig. 3, but different in your computer), and put a colon immediately after it – but *don't* press return just yet! The colon tells the computer that you are going to enter some new numbers in here. Type the following four numbers: 08 00 00 11, and press return. Type control-C to get back into Basic, and list line zero. It will now read: 0 LOMEM: 3072.

Copy And Flip Subroutine

The workhorse of this program is the subroutine at line 60. It does the job of copying what is on graphics page one to page two. The display on page one is not erased by the copying, and will stay there until your program changes it. Here's how the page-flipping routine works: While your subroutines are drawing or erasing on page one, the viewer is looking at a finished picture on page two. When the drawing on page one is complete, the viewer is switched to page one where he sees the next step in the movement of the figure. He hasn't seen any of the drawing or erasing, so it only appears that the figure has made a slight shift in position. While the viewer is looking at page one, the computer is busy copying page one to page two – invisibly. The subroutine then switches to page two, but the viewer is unaware that anything has happened since page two is now an exact copy of page one. Now, while the viewer is looking at page

13 key numeric KeyPad, essential to business applications.

KeyPad™

Bar code reader available with software to read and print various types of bar codes.

BarWand™

A great programmer's aid, this features single-key entry and customized key functions.

SoftKey™

APPLE* Peripherals From Your Local Dealer

ADVANCED BUSINESS TECHNOLOGY, INC.
12333 Saratoga – Sunnyvale Rd.
Saratoga, California 95070
Telephone: 408-446-2013

*Trademark of APPLE COMPUTER, INC.

★★★★★★★★★★★★★★★★

Bugs in your Apple?

DDT

Disco-Tech's Disc Drive Timer program

zaps disc drive problems!

Analyze disc drive motor speed on a routine basis with an adjustable real-time speedometer. Accurate to one-tenth of one r.p.m. out of 300 r.p.m.

Fine-tune disc drive motor speed yourself. All you need is DDT, two screwdrivers, and five minutes' time.

$29.95 Postpaid
California residents add 6% tax
Diskette & complete manual.

Engineering Business
Architecture Utilities
Surveying

Also available for TRS-80 Model I.

To order or for more information, write or call

disco tech T.M. microcomputer products a division of
Morton Technologies, Inc.
600 B Street
P.O. Box 11129 • Santa Rosa, CA 95406
707/523-1600
Dealer inquiries invited

★★★★★★★★★★★★★★★★

two, the cycle repeats and a new drawing is begun "behind the scenes" on page one.

The POKE-16300,0 at the beginning of subroutine 60 causes page one to be displayed, showing the viewer the drawing you made while he was looking at page two. The next six pokes specify that it is page one of graphics that is to be copied to page two. The CALL-468 does the actual copying. The last Poke, POKE-16299,0 causes page two to be displayed again, and the subroutine returns to the main program so that the next step in the drawing can be made out of sight on page one.

The speed of the animation can be slowed down if necessary with delay loops such as those in lines 50, 51, and 52. For large or very complicated drawings delay loops won't be necessary, but for a small object which is quickly drawn you may want them.

Main Program

The main program is found from line 1000 to 2000. You can see that it consists mostly of GOSUB's. There is a FOR...NEXT loop, and several X = X + 1 commands, which cause the figure to move to the right. Line 1000 clears the graphics and text display and calls the page-flip routine so that the viewer will be looking at a blank screen for a few microseconds while the first drawing is made. Lines 1010 and 1020 draw a sidewalk and place the girl on the sidewalk at the left of the screen, standing still (drawing A). Note that X was set to zero to place her at the left edge, and Y was set to 10 to put her down on the sidewalk. At the end of line 1020 the page flip subroutine is called, followed by a delay subroutine, so that the viewer looks at the standing girl for a brief time before she begins to walk. Lines 1030 through 1050 contain a FOR...NEXT loop during which the drawings B, C, D, A will be made seven times. (The loop starts at B since A was made before entering the loop, and it was desirable to finish with the girl in standing position). Note that the first GOSUB in line 1030 erases the girl. X is then incremented by one before the next drawing is made, so that she will appear to have shifted one space to the right. Each time a new drawing is made it must be erased before incrementing X and making the next drawing, or the figure will leave a trail of itself behind on the screen. In this example program the girl will move across the screen from left to right one square at a time. The last step of the program is to do a POKE-16300,0 at the end, so that the viewer will be left on page one when the program ends.

Final Hints

When you are debugging your program and using the page-flipping subroutine, you may occasionally hear the ominous "syntax error" beep, but be unable to see a message on the screen, and also see no cursor. This probably means you have been caught on page two, and the error message that stopped your program and left you stranded on page two is being displayed on page one. Just type POKE-16300,0 (return), and you will see page one displayed, where the error message and cursor will be visible.

It is hoped that you have a lot of fun with animating your graphics routines, despite the extra effort involved. The same general methods apply to Applesoft programs, but different "tricks" are required to gain access to page two and to do the memory move required. These "tricks" will be described in a future article.

```
>LIST
  0 LOMEM:3072
 10 POKE -16300,0: TEXT : CALL
     -936: GR : GOTO 1000
 49 REM  ** DELAY ROUTINES **
 50 FOR J=1 TO 50: NEXT J: RETURN

 51 FOR J=1 TO 1000: NEXT J: RETURN

 52 FOR J=1 TO 2000: NEXT J: RETURN

 59 REM  ** PAGE FLIPPER **
 60 POKE -16300,0: POKE 60,0: POKE
    61,4: POKE 62,255: POKE 63,
    7: POKE 66,0: POKE 67,8: CALL
    -468: POKE -16299,0: RETURN

 98 REM  ** GRAPHICS FOR GIRL **
 99 REM  ** DRAW TOP HALF **
100 COLOR=8: HLIN X+3,X+4 AT Y:
     VLIN Y,Y+2 AT X+3: PLOT X+
    2,Y+2
105 COLOR=13: VLIN Y+1,Y+2 AT X+
    4: COLOR=6: VLIN Y+3,Y+6 AT
    X+3: VLIN Y+3,Y+6 AT X+4
110 COLOR=3: VLIN Y+7,Y+8 AT X+
    3: VLIN Y+7,Y+8 AT X+4: HLIN
    X+2,X+5 AT Y+9: RETURN
119 REM  ** STANDING LEGS **
120 COLOR=13: VLIN Y+10,Y+11 AT
    X+3: COLOR=8: HLIN X+3,X+4 AT
    Y+12: RETURN
129 REM  ** STEP FORWARD **
130 COLOR=13: VLIN Y+10,Y+11 AT
    X+3: PLOT X+4,Y+10: PLOT X+
    5,Y+11: COLOR=8: HLIN X+3,X+
    4 AT Y+12: PLOT X+6,Y+12: PLOT
    X+7,Y+11: RETURN
139 REM  ** MIDDLE OF STEP **
140 COLOR=13: HLIN X+3,X+4 AT Y+
    10: PLOT X+2,Y+11: PLOT X+5
    ,Y+11: COLOR=8: HLIN X+2,X+
    3 AT Y+12: HLIN X+5,X+6 AT
    Y+12: RETURN
149 REM  ** END OF STEP **
150 COLOR=13: VLIN Y+10,Y+11 AT
    X+4: PLOT X+3,Y+10: PLOT X+
     2,Y+11: COLOR=8: VLIN Y+11,
     Y+12 AT X+1: HLIN X+4,X+5 AT
     Y+12: RETURN
 159 REM  ** ERASE GIRL **
 160 COLOR=0: FOR J=X+1 TO X+7: VLIN
     Y,Y+12 AT J: NEXT J: RETURN
 999 REM  ** MAIN ROUTINE **
1000 GR : CALL -936: GOSUB 60: TAB
     14: PRINT "GIRL WALKING"
1010 COLOR=5: HLIN 0,39 AT 23
1020 X=0:Y=10: GOSUB 100: GOSUB
     120: GOSUB 60: GOSUB 51
1030 FOR I=1 TO 7: GOSUB 160:X=X+
     1: GOSUB 100: GOSUB 130: GOSUB
     60: GOSUB 50
1040 GOSUB 160:X=X+1: GOSUB 100:
      GOSUB 140: GOSUB 60: GOSUB
     50: GOSUB 160:X=X+1: GOSUB
     100: GOSUB 150: GOSUB 60: GOSUB
     50
1050 GOSUB 160:X=X+1: GOSUB 100:
      GOSUB 120: GOSUB 60: GOSUB
     50: NEXT I
1060 CALL -936: TAB 17: PRINT "THE EN
     D": POKE -16300,0: GOSUB 52
     : GOSUB 160: COLOR=0: HLIN
     0,39 AT 23
2000 TEXT : CALL -936: POKE -16300
     ,0: END
```
©

SOFTWARE HOLDUP?

Photo: Dimmick's Doubles

TIRED OF HIGH PRICES & NO SERVICE?

Quality Software
Apple II - TRS-80 - TI 99/4 - Apple III

Creative Discount Software
256 South Robertson, Suite 2156
Beverly Hills, CA 90211

CALL for our Catalog **TOLL FREE**
Operator 831

800-824-7888
Alaska/Hawaii 800-824-7919
California 800-852-7777

HUNTINGTON COMPUTING

PROGRAMS LISTED BELOW ARE ON DISK:

Sabotage	$24.95	now	$21.19
HI-RES Soccer	$29.95	now	$25.39
Pulsar II	$29.95	now	$25.39
Orbitron	$29.95	now	$25.39
Autobahn	$29.95	now	$25.39
Gamma Goblins	$29.95	now	$25.39
EPYX (Automated Simulations) SPECIAL — While they last: Ryn/Morloc/Rigel all three			$39.99
Galaxy Wars	$24.95	now	$22.99
Super Scribe	$129.95	now	$110.39
Galactic Empire	$24.95	now	$21.20
Galactic Trader	$24.95	now	$21.20
Galactic Revolution	$24.95	now	$21.20
DB Master	$229.00	now	$194.59
Aristotle's Apple	$34.99	now	$29.99
Micro Memo	$39.95	now	$35.99
The Data Factory	$150.00	now	$124.00
Mission Asteroid	$19.95	now	$17.99
Galactic Attack	$29.95	now	$25.39
Wizardry	$39.95	now	$33.99
Bright Pen	$49.95	now	$42.49
Star Mines	$29.95	now	$25.39
PFS (Personal Filing System)	$95.00	now	$80.69
PFS: Report	$95.00	now	$80.69
Warp Factor	$39.95	now	$33.99
Vinyl holders for 20 disks in beautiful deluxe padded binder			$13.99
DOS Tool Kit	$75.00	now	$69.00
Microsoft Adventure	$29.95	now	$26.09
Olympic Decathlon	$24.95	now	$21.99
Compu-Math: Arithmetic	$49.95	now	$44.95
CRAE 2.0	$24.95	now	$21.99
MCAT 2.0	$19.95	now	$17.99
Cartels and Cutthroats	$39.95	now	$33.99
Operation Apocalypse	$59.95	now	$50.99
Torpedo Fire	$59.95	now	$50.99
Tank Attack	$20.00	now	$16.99
Modifiable Database II	$150.00	now	$127.49
TG Game Paddles	$39.95	now	$33.99
TG Joy Stick	$59.95	now	$50.99
The Tarturian	$24.95	now	$21.99
The Prisoner	$29.95	now	$26.99
The Wizard & The Princess	$32.95	now	$28.99
Flight Simulator (disk)	$34.95	now	$29.50
Star Cruiser	$24.95	now	$20.99
Space Album	$39.95	now	$34.50
Odyssey	$29.95	now	$25.50
Micro League Baseball	$24.95	now	$22.99
Sargon II	$34.95	now	$29.70
Program Line Editor	$40.00	now	$34.99
Space Eggs	$29.95	now	$25.39
Mystery House in French			$22.99
Z-80 Softcard	$349.00	now	$299.00
Videx 80-Col. Board	$350.00	now	$299.00
3D Super Graphics	$39.95	now	$35.99
Compu-Math I or II	$40.00	now	$34 ea.
HI-RES Cribbage	$24.95	now	$21.99
Phantom Five	$29.95	now	$25.49
Star Gazer's Guide	$30.00	now	$25.39
Lords of Karma	$20.00	now	$16.99
Apple PIE & Formatter (Reg. $129.95)		Special	$99.99
The Book of Software	$19.95	now	$17.99
Versa Writer Expansion Pac-I	$39.95	now	$33.99
The Linguist	$40.00	now	$33.99
Apple II Users Guide	$14.95	now	$12.99
Oh Shoot!			$19.99
Serendipity Statistics or Gradebook	$169.00	now	$143.59
Bridge Partner (disk)	$24.95	now	$21.19
ABM (Muse)	$24.95	now	$22.49
Data Plot (Muse)	$59.95	now	$52.99
Computer Conflict	$39.95	now	$35.99
Computer Air Combat	$59.95	now	$52.99
The Temple of Apshai	$39.95	now	$33.95
Super-Text II	$150.00	now	$128.99
Magic Window	$99.95	now	$85.00
Request	$225.00	now	$191.19
Space Raiders	$29.95	now	$25.39
Thinker	$495.00	now	$420.69
Super Kram	$175.00	now	$148.69
Savage Island I (disk)	$20.95	now	$17.79
Savage Island II (disk)	$20.95	now	$17.79
Thrilogy	$24.95	now	$21.19
Zork	$39.95	now	$33.99
Howardsoft Real Estate Analyzer	$150.00	now	$129.99
Super Disk Copy-Sensible	$30.00	now	$25.49
Fastgammon	$24.95	now	$21.20
Astro Apple	$20.00	now	$17.45
Akalabeth	$34.95	now	$29.99

Use Our new 800 Number (except Calif.) 800-344-4111

Use Our new 800 Number (except Calif.) 800-344-4111

MEMOREX DISKS 10 for $24.99

VISA

master charge

We take MasterCard or VISA (include card # and expiration date). California residents add 6% tax. Include $2.00 for postage. Foreign and hardware extra. Send for free catalog. Prices subject to change.

HUNTINGTON COMPUTING,
2020 Charles Street
Post Office Box 787
Corcoran, California 93212

SUPER DISCOUNTS
apple® SOFTWARE

Order by Phone (209) 992-5411

Oscilloscope

Rob Smythe
Ontario, Canada

Here is a program for physics teachers that makes good use of the Apple's high resolution graphics.

Unless your school's equipment is better than mine, you probably find it tricky to demonstrate waveforms in class. Stabilizing the pattern on an oscilloscope can require painstaking adjustment when the frequency of the inputted sound is changed. You wish to show how the shape of the wave is altered as overtones are added, but text-books don't show enough. Demonstrating the effect of different separations in the frequencies of two notes requires diagrams, tedious to produce.

With this program you can show effects of varying amplitude and frequency on sine waves, add up to five overtones (each with their own amplitude) and show the resultant wave pattern for up to six different notes. This last facility is useful for demonstrating the cause of beats.

When you run this Applesoft program you will be presented with a table showing that there are no notes presently in memory and a menu prompting you for single Keystroke selection of commands. Touch 1, 2, 3, 4, 5 or 6 and you will be able to set the amplitude and frequency of a note. Enter as many notes as you wish, or change them one by one. Touch P to plot the resultant waveform. After the oscilloscope pattern is drawn and you have finished studying it, return to the menu by pressing any key.

Touching S will enable you to alter the plotting speed, which is initially set at 4. This determines the increment along the x-axis (time axis) between plotted points. When using frequencies over about 500 Hz, you might have to set speed at 1 or 2 (because at coarser settings significant change to the shape might occur between points and be missed: try 800 Hz at speed 4 and speed 1 to see this).

To clear all notes from the table, touch C and confirm with a Y.

Try notes of amplitude 10 to 20 in a frequency range of 100 to 500. Create a complicated note using all overtones, with amplitudes 10 or less (so that you don't go off the top of the "scope"). Beat patterns look nice when you play notes of frequency 1000 and 1050 together.

The Program:

1000's	**print table and menu routine**
1030	**format numbers in display**
1100	**wait for single keystroke input**
1110	**input data**
1120 on	**process data and reject invalid input**
2000's	**plot routine**
2000-2110	**draw axes**
2150-2160	**pick X value in radians**
2170	**sum the waves**
2190	**scale X and Y to fit screen**
2200	**check for off scale values**
2210	**plot**
3000's	**subroutine to check that points are not off scale**

Variables (in order of appearance)

G$	**bell**
SP	**speed (1 = slow to 5 = fast)**
I	**counter, local pointer**
A$	**local input variable**
AMP(I)	**amplitude of the I-th note**
FR(I)	**frequency of the I-th note careful: don't use FRE(I)**
F(I)	**frequency after scaling for plotting**
TIME	**a measure of length of X-axis**
S	**scaling factor**
J	**loop counter**
X	**horizontal coordinate of point**
Y	**vertical coordinate of point**

Suggested Modifications:

1. Very small changes are required to allow for more overtones.

2. Changing TIME in line 2120 will allow for a different range of suitable frequencies. You might add TIME input to the menu, so that beats can be shown effectively with frequencies that are very close together.

3. Of course, adding routines which would produce the sound of the note you have created on the Apple's speaker would make this program tremendously useful. Any volunteers?

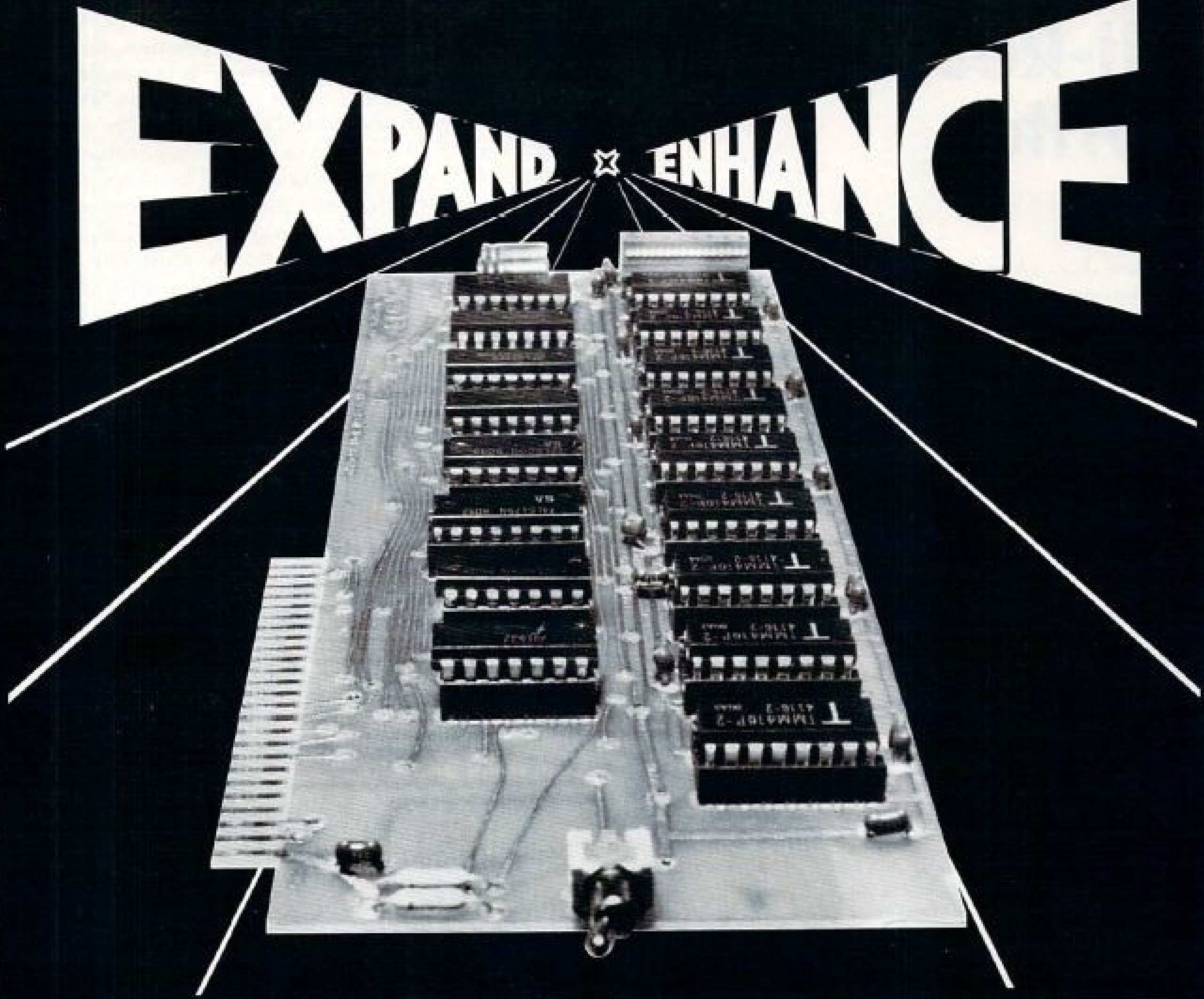
```
]LIST

10  REM   OSCILLOSCOPE

          R.M. SMYTHE
          1522 RUSHOLME CRES.
          BURLINGTON, ONTARIO
          CANADA
```

```
20  REM

       COPYRIGHT (C) 1981
       BY SOFTWARE UNLIMITED

50 G$ =  CHR$ (7): REM ERROR BEEP

100 SP = 4: REM  PLOTTING SPEED
    FROM 1 (SLOW=MOST ACCURATE)
    TO 5
997 :
998  REM  DATA INPUT
999 :
1000  TEXT : HOME
1010  PRINT "     NOTE     AMP     F
    REQ": PRINT
1020  FOR I = 1 TO 6: PRINT  TAB(
    7);I;"      ";
1030 A$ =  RIGHT$ ("       " + STR$
    (FR(I)),6): IF AMP(I) < 10 THEN
     PRINT " ";
1040  PRINT AMP(I);"  ";A$
1050  PRINT : NEXT I
1060  PRINT : PRINT : PRINT "SPEE
    D - ";SP
1070  VTAB 21
1080  PRINT "CHANGE NOTE: 1/2/3/4
    /5/6    PLOT: P"
1090  PRINT "CLEAR NOTES: C    EXI
    T: E   SPEED: S"
1100  POKE  - 16368,0: WAIT  - 16
    384,128
1110  GET A$:I =  VAL (A$): IF I >
    6 THEN  PRINT G$: GOTO 1000
1120  IF I = 0 THEN 1180
1130  VTAB 21: CALL  - 958: PRINT
    "NOTE ";I;": ";: INPUT "AMPL
    ITUDE (1-10) ";A$:AMP(I) =  VAL
    (A$): IF AMP(I) = 0 THEN 113
    0
1140  IF AMP(I) > 20 THEN  PRINT
    G$;: GOTO 1130
1150  PRINT  TAB( 9);: INPUT "FRE
    QUENCY - ";FR(I): IF FR(I) <
    0 OR FR(I) > 99999 THEN  PRINT
    G$;: VTAB 22: CALL  - 868: GOTO
    1150
1160 F(I) = FR(I) / 27.75
1170  GOTO 1000
1180  IF A$ = "E" THEN  END
1190  IF A$ = "P" THEN 2000
1200  IF A$ = "C" THEN 1240
1210  IF A$ <  > "S" THEN  PRINT
    G$: GOTO 1000
1220  VTAB 21: CALL  - 958: INPUT
    "ENTER SPEED (1-5) - ";SP: IF
    SP < 1 OR SP > 5 OR  INT (SP
    ) <  > SP THEN  PRINT G$: GOTO
    1220
1230  GOTO 1000
1240  VTAB 21: CALL  - 958: PRINT
    "CLEAR ALL NOTES IN MEMORY?
    (Y/N) ": GET A$: IF A$ <  >
    "Y" THEN 1000
1250  FOR I = 1 TO 6:F(I) = 0:FR(
    I) = 0:AMP(I) = 0: NEXT : GOTO
    1000
1997 :
1998  REM  PLOTTING ROUTINE
1999 :
2000  HOME
2010  VTAB 24
2020  HGR
2030  HCOLOR= 3
2040  HPLOT 0,80 TO 279,80
2050  HPLOT 0,16 TO 0,143
2060  FOR I = 0 TO 279 STEP 70
2070  HPLOT I,78 TO I,82: HPLOT 2
    79,78 TO 279,82
2080  NEXT I
2090  FOR I = 16 TO 144 STEP 16
2100  HPLOT 0,I TO 4,I
2110  NEXT I
2120 TIME = 400
2130 S = 280 / TIME
2140  HPLOT 0,80
2150  FOR I = 0 TO TIME STEP SP
2160 X = I * 3.14159 / 180
2170 Y = 0: FOR J = 1 TO 6:Y = AM
    P(J) / 5 *  SIN (F(J) * X) +
    Y: NEXT J
2180 Y = 80 - Y * 16
2190 X = I * S
2200  GOSUB 3000
2210  HPLOT  TO X,Y
2220  NEXT I
2230  POKE  - 16368,0: WAIT  - 16
    384,128
2240  GET A$
2250  GOTO 1000
2997 :
2998  REM  SUBROUTINE CHECK RANGE

2999 :
3000  IF X < 0 THEN X = 0
3010  IF X > 279 THEN X = 279
3020  IF Y < 0 THEN Y = 0
3030  IF Y > 159 THEN Y = 159
3040  RETURN
```

©

Apple Authors

COMPUTE! is looking for applications articles aimed at beginners and intermediate programmers. We're specifically interested in programming hints, tutorials, articles written to help users get more out of their machine.



16K RAM EXPANSION BOARD FOR THE APPLE II* $195.00

The Andromeda 16K RAM Expansion Board allows your Apple to use RAM memory in place of the BASIC Language ROMs giving you up to 64K of programmable memory. Separate Applesoft* or Integer BASIC ROM cards are no longer needed. The 16K RAM Expansion Board works with the Microsoft Z-80 card, Visicalc, DOS 3-3, Pascal, Fortran, Pilot, and other software. A switch on the card selects either the RAM language or the mainboard ROMs when you reset your Apple.

The Andromeda 16K RAM Expansion Board has a proven record for reliability with thousands of satisfied customers.

Now with One Year Warranty.

*Apple II and Applesoft are trademarks.

ANDROMEDA
INCORPORATED
P.O. Box 19144
Greensboro, NC. 27410
919 852-1482

Distributed By:

COMPUTER DATA SERVICES
P.O. Box 696
Amherst, NH. 03031
603 673-7375

The Apple Hi-Res Shape Writer

Doug Hennig
Dallas, TX

These days a lot of people are writing their own games or applications software rather than paying for someone else's labor. After all, designing and writing software really is the best part about owning your own microcomputer. Games especially are fun to design because they tax not only your programming skills, but your imagination as well.

Many of today's popular computer games are of the "arcade" type where you are required to fire missles to destroy any number of different kinds of objects. Usually these objects are drawn and manipulated on the high resolution screen using hi-res shape tables that are stored in memory to define the object. Other types of games also use hi-res shape tables, including a number of recently developed high resolution adventure games. As you have no doubt already observed, the uses of shape tables in graphics programs is almost limitless. However, that is where the problem arises.

Anyone who has ever designed a high resolution shape table knows that it is not a lot of fun to do. You must draw the shape dot by dot, convert each dot into a number, convert the numbers into bytes by an obscure process, type in the long list of bytes, and hope that you have not made an error somewhere along the way (I invariably do!). After several attempts at such nonsense, I thought that there must be an easier way. Why not do just the fun part – drawing the shape – and leave the Apple to do the hard part? Thus the Apple Hi-Res Shape Writer was born.

The Apple Hi-Res Shape Writer allows you to draw any shape on the low resolution screen using the game paddles and then convert the lo-res shape into a high resolution shape table. You will soon see your graphics ideas take shape (no pun intended) quickly and painlessly. Create space ships, alien creatures, even new and exotic character sets.

Operation

Operation of the Apple Hi-Res Shape Writer is simple. First you tell the program how many shapes you want in the shape table (the maximum is ten). It will then draw a dark blue background in low resolution graphics and ask you for the number of vertical and horizontal elements in the shape that is to be drawn next; since the low resolution screen is 40x40, these are the limits for the shape. A black area that is the size of your shape will appear, which can be filled in as you wish. The location of the current plotting position, indicated by a flashing light blue "cursor," is controlled by the game paddles. To plot a point, press "P" and to erase a previously plotted point, press "E" (note that the REPT key can be used along with either of these to give "speed" drawing). The shape does not have to be drawn in any particular order; you can "doodle" if you want, trying different designs and erasing the parts that you do not want. Once the shape is drawn to your satisfaction, press "S" to construct the shape table. After the table is done, you may see the high resolution shape before starting on the next shape. Once you have drawn the desired number of shapes, you have the option of saving the entire shape table on disk.

Hi-res shapes will appear "skinnier" than the lo-res shapes drawn because low resolution blocks are rectangular rather than square. However, a little practice will allow you to easily visualize how your hi-res shape will look and to plan the lo-res shape accordingly.

Since the hi-res shape table is stored in the second page high resolution buffer (starting at 16384 or $4000), The Apple must have at least 24K and Applesoft in ROM.

The Program

There are few "fancy tricks" used in this program. If you are not familiar with the way shape tables are created or used, you should read Chapter Nine of the latest Applesoft manual before trying to follow how this program works. To help explain its operation, I have included a list of major variables and subroutines used in the program, with comments about the uses of each.

Variables Used (In Order of Appearance)

A$ Used for all input from the user.

SH The number of shapes to be drawn (a maximum of ten). This number is POKEd into the first two locations of the shape table.

BASE The starting address of the data for the current shape. It is initially set to 16384 + 2*SH + 2 because the table starts at 16384 and the table index consists of two bytes for the number of shapes and two bytes to point to each shape.

NU The number of the current shape.

TABLE () Holds the plot status of every point in the shape area of the screen. The status is based on the following system:

move right (no plot) = 1	plot and move right = 5
move down (no plot) = 2	plot and move down = 6
move left (no plot) = 3	plot and move left = 7

BLACK, AQUA, RED Hold the values of the corresponding low resolution colors.

COL A dual purpose variable: first it holds the number of horizontal elements in the shape and later it is used as a loop counter for the columns.

C1, C2 Contain the horizontal low resolution screen limits.

R1, R2 Contain the vertical low resolution screen limits.

X, Y Represent the coordinates of the current plot position.

OLD Holds the color of the screen under the "cursor" (black if nothing was plotted, red if a point was plotted).

MOVE Stores the plot value: either move right (1) or move left (3), depending on which row is being scanned (see the variables START, LAST).

DOWN Stores the plot value to move down (2).

START, LAST Hold the loop limits for the column counter. The first row is scanned left to right, the second right to left, and so on (always moving down one row when the end of the current row is reached), so START and LAST are switched at the end of each row.

INC Another dual purpose variable: initially, it holds the step value for the column loop (1 means scanning left to right and -1 means scanning right to left); later it is used to hold the number of bytes in the current shape.

B Holds the octal representation of the current byte to be put into the shape table. Octal is used because in hi-res shape tables, eight possible moves are allowed (see the Applesoft manual).

W, X, Y, Z Store intermediate results in the conversion of octal to decimal. X also holds the final (decimal) result.

D Holds the length of the shape table.

Sections And Subroutines Used

10–37	Introduction.
40–400	Print instructions.
410–450	Initial shape table setup.
460–500	Variable initialization.
510–700	Low resolution screen setup.
710–830	Draw the lo-res shape.
840–1130	Create the hi-res shape table.
1140–1240	Display the hi-res shape.
1250–1500	Save the shape table on disk.
11000–11080	Input user response (avoids the problems of using INPUT).
12000–12100	Print heading.
13000–13040	Go on to next screenful.

Changes And Modifications

A number of changes to the program are possible. The colors of the background, the flashing "cursor," and plotted points may be changed to suit your taste by changing the values in lines 490 and 530 (AQUA is the color of the "cursor" and RED is the color of a plotted point). To allow more shapes per table (up to a reasonable limit, of course), change the upper limit in line 425.

Perhaps the most important modification would concern the speed of execution. Creation of the shape table can take up to several minutes for large shapes because of the amount of data involved. If the time involved seems unreasonable to you, perhaps a machine language version of lines 840 to 1130 would be in order. Since I am not proficient in machine language, I chose to let Basic do the job for me, but I am sure that some enterprising soul can come up with a faster version. If you want to tackle this problem, feel free to contact me with any questions that you have about the program.

Using Shape Tables With Your Own Programs

There are two ways that you can use a shape table in your own program. The program can read in the table from disk (using BLOAD) or the program can POKE in the values for the table from data in DATA statements. The first method is obviously easier to program, but you must ensure that the disk containing the shape table is inserted in the drive or the user will get a nasty DOS error message.

The question with the second method is: how do I convert all those bytes in memory into numbers in DATA statements? One way (the hard way) is to write a short program which reads the shape table one byte at a time, prints the value, and lets you write it down before going on to the next one. Then you have to type all the values into DATA statements and POKE them into memory. The easier way is to EXEC a text file which will do all that for you. Listing 2 contains a program which will set up such a text file. To use this, set the variable LINE in line 90 to the line number that you want the POKE routine to start at, set the variable B in line 90 to the last memory location of the shape table (lines 1460 and 1470 in the Apple Hi-Res Shape Writer print this value for you), RUN the program and EXEC the text file. For example, for a shape table that the computer tells you ends at 17000, type (in the direct mode):

```
LOAD POKE WRITER
90 LINE = 5000 : B = 17000
RUN
```

Now just load your program, EXEC POKE ROUTINE, and you have added a routine, starting at line 5000 in this example, that will POKE your shape table into memory every time the program is run.

I hope that you enjoy Apple Hi-Res Shape Writer. I have found it extremely useful in designing some of my own games and educational software, and I am sure that you will find many uses for it too!

```
?SYNTAX ERROR
]LIST

10  REM ********************************
12  REM ** APPLE HI-RES SHAPE WRITER **
14  REM **       BY DOUG HENNIG       **
20  REM ********************************
25  HIMEM: 16384
30  GOSUB 12000
35  PRINT "DO YOU WANT INSTRUCTIONS? ";: GOSUB 11000
37  IF A$ = "N" THEN  HOME : GOTO 400
40  VTAB 10: CALL  - 868: HTAB 3
50  PRINT "THIS PROGRAM WILL ALLOW YOU TO CREATE"
60  PRINT "UP TO TEN DIFFERENT SHAPE TABLES FROM"
70  PRINT "DESIGNS THAT YOU HAVE DRAWN ON THE"
80  PRINT "SCREEN IN LOW-RESOLUTION GRAPHICS."
90  VTAB 15: HTAB 3
100  PRINT "THE PROGRAM WILL ASK FOR THE NUMBER"
110  PRINT "OF HORIZONTAL AND VERTICAL ELEMENTS "
120  PRINT "THAT YOU WANT. THESE NUMBERS MUST BE"
130  PRINT "BETWEEN 1 AND 40."
140  GOSUB 13000
150  HTAB 3
160  PRINT "A BLUE 'CURSOR' WILL BE DISPLAYED TO"
170  PRINT "INDICATE THE CURRENT PLOT POSITION. USE"
180  PRINT "THE GAME PADDLES TO MOVE THE CURSOR TO"
190  PRINT "THE DESIRED LOCATION AND PRESS ";: INVERSE : PRINT "P";: NORMAL
   : PRINT " TO"
200  PRINT "PLOT AT THE CURRENT POSITION; A PLOTTED"
210  PRINT "POINT IS INDICATED BY A RED SQUARE. TO"
220  PRINT "ERASE A PLOTTED POINT, PRESS ";: INVERSE : PRINT "E";: NORMAL
   : PRINT ". TO"
230  PRINT "BEGIN CREATION OF THE SHAPE TABLE,"
240  PRINT "PRESS ";: INVERSE : PRINT "S";: NORMAL : PRINT ". "
250  GOSUB 13000
260  HTAB 3
270  PRINT "THE SHAPE TABLES WILL BE STORED IN"
280  PRINT "THE HIGH-RESOLUTION SECONDARY PAGE AREA"
290  PRINT "(STARTING AT LOCATION 16384, OR $4000)."
300  PRINT "YOU WILL BE GIVEN THE OPTION TO SAVE "
310  PRINT "THE SHAPE TABLE ON DISK."
320  PRINT : HTAB 3
330  PRINT "TO USE THE SHAPE TABLE IN A PROGRAM,"
340  PRINT "SIMPLY USE A 'BSAVE' COMMAND WITHIN"
350  PRINT "THE PROGRAM. NOTE THAT THE HIGH-"
360  PRINT "RESOLUTION SECONDARY PAGE WILL BE"
370  PRINT "UNAVAILABLE FOR USE."
390  GOSUB 13000
400  POKE 34,0
410  PRINT "HOW MANY SHAPES WILL THERE BE? ";: GOSUB 11000
420 SH =  INT ( VAL (A$))
425  IF SH < 1 OR SH > 10 THEN  VTAB 10: CALL  - 958: GOTO 410
430  POKE 16384,SH: POKE 16385,0: REM PUT NUMBER OF SHAPES INTO START OF T
     ABLE INDEX
440 BASE = 16384 + 2 * SH + 2
450  POKE 232,0: POKE 233,64: REM TELL APPLE WHERE SHAPE TABLE IS
460 NU = 0
480  DIM TABLE(1600)
490 BLACK = 0:AQUA = 6:RED = 1
```

```
500 NU = NU + 1: IF NU > SH THEN 1250
510  GR
520  HOME : VTAB 21
530  COLOR= 2
540  FOR I = 0 TO 39: VLIN 0,39 AT I: NEXT I: REM DRAW BACKGROUND
550  PRINT "NUMBER OF HORIZONTAL ELEMENTS - ";: GOSUB 11000
560 COL =  INT ( VAL (A$))
570  IF COL < 1 OR COL > 40 THEN  VTAB 21: CALL  - 958: GOTO 550
580  HOME : VTAB 21
590  PRINT "NUMBER OF VERTICAL ELEMENTS - ";: GOSUB 11000
600 ROW =  INT ( VAL (A$))
610  IF ROW < 1 OR ROW > 40 THEN  VTAB 21: CALL  - 958: GOTO 590
620 C1 =  INT ((40 - COL) / 2):R1 =  INT ((40 - ROW) / 2)
630 X =  INT (COL / 2 -  INT (COL / 2) + 0.5):Y =  INT (ROW / 2 -  INT (RO
    W / 2) + 0.5)
640 C2 = 39 - C1 - X:R2 = 39 - R1 - Y
650  COLOR= BLACK: FOR I = C1 TO C2: VLIN R1,R2 AT I: NEXT I: REM CLEAR SP
    ACE FOR SHAPE
660  HOME : VTAB 21
670  INVERSE : PRINT "P": PRINT "E": PRINT "S": NORMAL
680  VTAB 21: HTAB 3: PRINT "- PLOT POINT"
690  VTAB 22: HTAB 3: PRINT "- ERASE POINT"
700  VTAB 23: HTAB 3: PRINT "- CREATE SHAPE TABLE"
710 X = C1:Y = R1
720 OLD =  SCRN( X,Y)
730  COLOR= AQUA: PLOT X,Y: REM FLASH CURSOR
740  FOR I = 1 TO 100: NEXT I
750  COLOR= OLD: PLOT X,Y
760 X =  INT ((C2 - C1 + 1) / 255 *  PDL (0)) + C1:Y =  INT ((R2 - R1 + 1)
     / 255 *  PDL (1)) + R1: REM GET NEW COORDINATES
770  IF X > C2 THEN X = C2: REM DON'T GO OUT OF BOUNDS
780  IF Y > R2 THEN Y = R2
790 KEY =  PEEK (49152):A =  PEEK (49168)
800  IF KEY < 128 THEN 720
810  IF KEY = 208 THEN  COLOR= RED: PLOT X,Y: REM PLOT POINT
820  IF KEY = 197 THEN  COLOR= BLACK: PLOT X,Y: REM ERASE POINT
830  IF KEY <  > 211 THEN 720
835  REM CREATE SHAPE TABLE
840  HOME : VTAB 21: PRINT "CREATING SHAPE TABLE"
850 MOVE = 1:N = 0:DOWN = 2
860 START = C1:LAST = C2:INC = 1
865  REM STARTING AT THE UPPER LEFT CORNER, SCAN BACK AND FORTH ACROSS ROW
     S
870  FOR ROW = R1 TO R2
880 : FOR COL = START TO LAST STEP INC
890 ::TABLE(N) = MOVE
900 :: IF COL = LAST THEN TABLE(N) = DOWN: REM IF END OF ROW GO DOWN
910 :: IF  SCRN( COL,ROW) = 1 THEN TABLE(N) = TABLE(N) + 4: REM A PLOT POI
     NT
920 ::N = N + 1
930 : NEXT COL
940 :TEMP = START:START = LAST:LAST = TEMP
950 MOVE = MOVE + INC * 2
960 :INC =  - INC
970  NEXT ROW
980 COL = 0:INC = 0
990 B = 0: FOR I = 0 TO 2: REM CONVERT MOVES TO BYTES
1000 :A =  INT (10 ^ I * TABLE(COL))
1010 :B = B + A
```

```
1020 : IF B > 199 THEN B = B - A: GOTO 1050
1030 :COL = COL + 1: IF COL = N THEN 1050
1040  NEXT I
1050 W =  INT (B / 100):X =  INT (B / 10)
1060 Y = X - W * 10:Z = B - X * 10
1070 X = 64 * W + 8 * Y + Z
1080  POKE BASE + INC,X:INC = INC + 1: REM PUT BYTE INTO TABLE
1090  IF COL <  > N THEN 990
1100  POKE BASE + INC,0:INC = INC + 1: REM END OF THIS SHAPE
1110  POKE 16384 + 2 * NU,BASE - 16384 - 256 *  INT ((BASE - 16384) / 256)
     : REM POKE POINTERS TO THIS SHAPE INTO INDEX
1120  POKE 16385 + 2 * NU, INT ((BASE - 16384) / 256)
1130 BASE = BASE + INC
1140  HOME : VTAB 21
1150  PRINT  CHR$ (7)"WANT TO SEE HI-RES SHAPE #"NU"? ";: GOSUB 11000
1160  IF A$ = "N" THEN 1250
1170  HGR : SCALE= 1: ROT= 0
1180  HCOLOR= 3
1190  DRAW NU AT 140,80
1200  HOME : GOSUB 13000
1240  GOTO 500: REM NEXT SHAPE
1250  REM   SAVE ON DISK
1260  GOSUB 12000
1270  VTAB 10: PRINT "DO YOU WANT TO SAVE THE SHAPE TABLE ON"
1280  PRINT "DISK? ";: GOSUB 11000
1290  IF A$ <  > "Y" THEN 1480
1300 D = BASE - 16384: REM  LENGTH OF TABLE
1340  PRINT : PRINT "FILE NAME - ";: GOSUB 11000
1350  IF  LEN (A$) > 30 OR  VAL ( LEFT$ (A$,1)) <  > 0 THEN  VTAB  PEEK (3
     7) - 1: CALL  - 958: GOTO 1340
1355  FOR I = 1 TO  LEN (A$): IF  MID$ (A$,I,1) = "," THEN  VTAB  PEEK (37
     ) - 1: CALL  - 958: GOTO 1340
1357  NEXT
1360  PRINT : PRINT "INSERT THE DATA DISK INTO THE DRIVE"
1380  PRINT "AND THEN PRESS ANY KEY."
1390 KEY =  PEEK (49152): IF KEY < 128 THEN 1390
1400 KEY =  PEEK (49168)
1410  PRINT : PRINT "SAVING SHAPE TABLE"
1415  ONERR  GOTO 5000
1420  PRINT  CHR$ (4)"BSAVE"A$",A$4000,L"D
1430  POKE 216,0
1460  PRINT : PRINT "THE LAST LOCATION IN THE SHAPE TABLE"
1470  PRINT "IS "BASE - 1"."
1480  POKE 34,0
1490  PRINT : PRINT "GOOD LUCK WITH YOUR NEW SHAPE TABLE!"
1600  END
4999  REM CONVERT DECIMAL TO HEX
5000  PRINT : PRINT "THERE WAS A DISK I/O ERROR.": POKE 216,0: END
10999  REM "INPUT" SIMULATOR
11000 A$ = ""
11010  GET B$
11020  IF B$ =  CHR$ (13) THEN  PRINT : RETURN : REM IF RETURN PRESSED GO
     BACK
11025  IF B$ =  CHR$ (21) OR B$ =  CHR$ (10) THEN 11010
11030  IF B$ <  >  CHR$ (8) THEN  PRINT B$;:A$ = A$ + B$: GOTO 11010
11040  IF  LEN (A$) = 0 THEN 11010: REM IF NO CHARS ENTERED IGNORE BACKSPA
     CE
11050  PRINT B$" "B$;
11060  IF  LEN (A$) = 1 THEN 11000
```

```
11070 A$ =  LEFT$ (A$, LEN (A$) - 1)
11080  GOTO 11010
11999  REM PRINTHEADING AND SET TEXT WINDOW
12000  TEXT : HOME
12010  VTAB 2: INVERSE
12020  FOR I = 1 TO 40: PRINT " ";: NEXT I: PRINT
12030  VTAB 4: HTAB 10: NORMAL
12040  PRINT "HI-RES SHAPE WRITER"
12050  HTAB 13: PRINT "BY DOUG HENNIG"
12060  VTAB 7: INVERSE
12070  FOR I = 1 TO 40: PRINT " ";: NEXT I: PRINT
12080  POKE 34,9
12090  NORMAL : VTAB 10
12100  RETURN
12999  REM NEXT SCREEN ROUTINE
13000  VTAB 23: HTAB 8
13010  INVERSE : PRINT "PRESS RETURN TO CONTINUE": NORMAL
13020 KEY =  PEEK (49152): IF KEY < 128 THEN 13020
13030 KEY =  PEEK (49168)
13040  HOME : RETURN

]
```

©

PROGRAMS AVAILABLE FROM COMPUTER HOUSE DIV.

FOR COMMODORE AND APPLE COMPUTERS

ACCOUNTING SERIES
- A/P, A/R, Job Cost and Job Estimating
- Payroll
- Checkwriter
- Inventory
- Mailing List

ATTORNEYS SERIES
- Financial
- Legal Accounting
- Legal Demo

CBM PROGRAMING AIDS SERIES
- Docu-Print
- F.E.T./Recover
- Screen Dump/Repeat
- Scrunch Plus
- Sof-Bkup
- Sorter
- Super-Ram
- Trace-Print
- Vari-Print

DESIGNERS/ENGINEERS SERIES
- Beams
- Bolt Circle
- Machine Part Quoting
- Machine Part Quoting Demo
- Spur Gears
- Trig/CircleTangent

POLITICAL SERIES
- Political Mailing List

REAL ESTATE SERIES
- Listings
- Financial with Record of Investments
- Mail/Phone List
- Checkwriter

APPLE PROGRAMING AIDS SERIES
- Scrunch

Prices & specifications subject to change without notice. Non-disclosure statement must be signed and returned before shipment.

NOTE: All of these programs are menu driven and prompt the user. Previous experience is not necessary, only familiarity with subject material. Instructions included with each program.

DEALERS WANTED

COMPUTER HOUSE DIV.
F.L.C., INC.
1407 Clinton Rd. Jackson, Mich. 49202
Telephone: (517) 782-2132

LOGICAL SOFTWARE, INC.

ANNOUNCES:

MAIL EXPRESS

A NEW MAIL LIST UTILITY FOR THE APPLE II.

- Up to 2,200 Names per File
- Sort by Company Name, Customer Name, City, State Zip
- Prints Return Addresses
- Merge up to 16 Files
- Easy User Definable Codes for City, State and Zip to Save Time and Disk Space

This is an easy to use professional quality mail list able to handle large or small files.

Introductory Price $49.95
$2.00 Postage & Handling

Logical Software, Inc.
P.O. Box 354
Farmington, MI 48024
(313) 474-8774

® Apple and Apple II are registered trademarks of Apple Computer Inc.

With the Hayes direct-connect Micromodem II,™ your Apple II can communicate by phone with the outside world. You can access information networks like The Source for a variety of business and personal applications, exchange programs with friends anywhere in North America, and even use your Apple II when you're away from your home or office.

Dependable. The Micromodem II is so dependable it comes with a *two-year* limited warranty. That's another reason why it's the largest selling direct-connect modem for Apple II computers.

Programmable. Automatic dialing and answering? Of course! We include programs on disk that dial phone numbers for you, send messages while you're away, and much more!

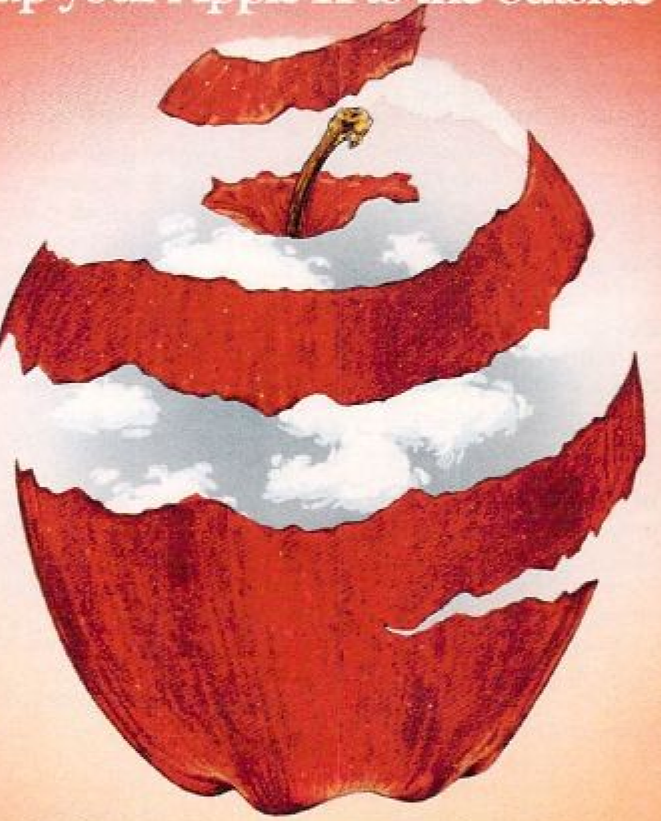
Complete. You get everything you need to communicate with other Bell 103 compatible modems at 110 or 300 baud. The serial interface is built-in, and we even include our FCC-approved Microcoupler™ that plugs directly into any modular telephone jack in the U.S. – you don't even need the phone!

S-100, too. The Hayes Micromodem 100 gives S-100 microcomputers all the advantages of our Micromodem II!

Put the outside world *inside* your computer with a data communications system from Hayes. Available at computer stores nationwide – call or write for the location nearest you. And don't settle for anything less than Hayes.

Hayes™

The Hayes Micromodem II opens up your Apple II to the outside world.



Hayes Microcomputer Products Inc. 5835 Peachtree Corners East, Norcross, Georgia 30092 (404) 449-8791

Micromodem II, Micromodem 100 and Microcoupler are trademarks of Hayes Microcomputer Products Inc.
†TM Apple Computer Inc. Micromodem II can also be used with the Bell & Howell computer. © 1981 Hayes Microcomputer Products Inc.

Apple Disk Motor Control

William W. Martin
Seven Valleys, PA

The purpose of this article is to demonstrate a method of software motor control operation for the APPLE II DISK system.

The original reason for adoption of this performance patch was to decrease the time required to load the data file for a TEXT EDITOR I normally use at work. This design can be used anytime maximum performance is desired. The improvement seen for the demonstration program is about 20 percent, but may be greater for TEXT FILES, especially when time is taken to process the data as it is input from the disk.

This performance patch stops the APPLE DOS from turning off the disk motor as normally done during a disk operation. This function can now be controlled by the user with a definite performance improvement without modifying the normal DOS skew parameters.

EXAMPLE: First, since we will be changing the instructions of the DOS, please use a scratch disk in case we destroy it.

Now, please perform these steps in the following order, from the immediate mode.

VERIFY ORIGINAL DATA..

1.'PRINT PEEK (-16834)' - DOS VERSION 3.2 or 3.2.1
OR
'PRINT PEEK (-16819)' - DOS VERSION 3.3
THE VALUE RETURNED SHOULD BE: 189

TO SET DOS PATCH..

2. 'POKE -16834,96' - DOS VERSION 3.2 OR 3.2.1
OR
'POKE -16819,96' - DOS VERSION 3.3

3. NOW TYPE 'CATALOG'

Notice that the disk is still spinning after completion of the 'CATALOG'.

4. TYPE 'CATALOG' AGAIN

Notice how the 'CATALOG' function is performed faster since the motor is already up to speed. Repeat step four to test this again.

TO RESTORE TO NORMAL..

5. 'POKE' -16834,189' - DOS VERSION 3.2 OR 3.2.1
OR
'POKE -16819,189' - DOS VERSION 3.3

This will restore the DOS back to its original value.

6. 'POKE -16152,0' - ALL DOS VERSIONS

This will turn the motor off.

NOTE: 'POKE -16151,0' - TURN DISK MOTOR ON
'POKE -16152,0' - TURN DISK MOTOR OFF

This same method can be used under program control to obtain the same results.

The Demo Program

After entering the program and verifying that it is correct, insert your scratch disk. Type RUN and observe how the DISK motor stops while loading normally, but continues to spin while loading the "TEST" file with the patch set.

Program Description

Lines 100–150 — These lines print the header, change the text window setting, and send the DOS command 'MON I,O,C'. There is a centering routine used here by setting 0$ to the desired output and then executing a GOSUB 970.

Lines 160–179 — This sets up the variable 'D' to a value (200 in this example) that is used to simulate a short delay which would normally be encountered if the input data is processed while being input.

Lines 180–310 — This section establishes the dummy T$ as 'ABCDEFGHIJKLMMNOPQRSTU VWXYZ' and stores it to the scratch disk 100 times.

Lines 320–500 — The 100 T-strings are now read back from file 'TEST' under the normal DOS. Notice how the motor turns on and off while reading the 'TEST' file. Each time the motor starts up again, some time is wasted while waiting for the disk to come to the proper speed.

Lines 510–760 — The 100 T-strings are again read back, but this time using our disk motor patch. Notice that with the patch, the disk motor no longer stops and always maintains proper speed. Besides the time improvement, it is probably a little easier on the APPLE power supply by reducing the on/off cycle for the motor.

NOTE: Line 590–610 — The ONERR GOTO at line 610 is necessary when using this patch because, without it, if a disk error is encountered, the program would break without restoring the original values and turning off the motor. Notice that the ONERR GOTO routine at line 950 insures that the original will be restored due to a disk error of some type.

Assembly Language Notes
What this routine does is replace an LDA ,X instruction that normally stops the disk motor with an RTS instruction. This tricks the DOS into thinking that it turned off the motor.

NOTE: I wish to give special thanks to Mr. E. L. Didion for the information to make this program possible.

```
100  REM   MOTOR CONTROL DEMO BY BILL MARTIN
110  :
120  TEXT : HOME : PRINT :O$ = "DOS MOTOR CONTROL DEMO": GOSUB 97
     0: PRINT :O$ = "BY BILL MARTIN": GOSUB 970
130 O$ = "--------------------------------------": GOSUB 970: POKE
     34,5: HOME
140 D$ =  CHR$ (4): PRINT D$;"MON I,O,C"
150 :
160 D = 200: REM    VARIABLE 'D' IS LOOP DELAY TO SIMULATE INPUT
     DATA PROCESSING
170 :
180 T$ = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
190  REM    SAVE T$ 100 TIMES
200 :
210  HOME : PRINT "PRESS A KEY TO SAVE 'TEST' FILE..";: CALL  - 7
     56: PRINT : HOME
220 O$ = "SAVING 'TEST' FILE": GOSUB 970:O$ = "------------------
     -----------------": GOSUB 970: POKE 34,8: HOME
230 :
240  PRINT D$;"OPEN TEST"
250  PRINT D$;"DELETE TEST"
260  PRINT
270  PRINT D$;"OPEN TEST"
280  PRINT D$;"WRITE TEST"
290  FOR N = 1 TO 100: PRINT T$: NEXT
300  PRINT D$;"CLOSE"
310 :
320  REM  READ BACK NORMAL
330 :
340  REM  TAKES ABOUT 51 SECONDS
350 :
360  HOME : PRINT "PRESS A KEY TO READ 'TEST' FILE NORMAL";: CALL
      - 756: PRINT : HOME
370  POKE 34,5: HOME :O$ = "READING 'TEST' FILE - NORMAL": GOSUB
     970:O$ = "------------------------------------": GOSUB 970: POKE
     34,8: HOME
380 :
390  PRINT D$;"OPEN TEST"
400  PRINT D$;"READ TEST"
410  FOR N = 1 TO 100
420  INPUT "";T$
430 :
440  REM  SIMULATE PROCESSING INPUT DATA
450  FOR C = 1 TO D: NEXT C
460 :
470  NEXT N
480  PRINT D$;"CLOSE"
490 :
```

```
500  PRINT : PRINT "NORMAL READ COMPLETE": PRINT  CHR$ (7): PRINT

510  POKE 34,5: HOME :O$ = "READING 'TEST' FILE - IMPROVED": GOSUB
     970:O$ = "------------------------------": GOSUB 970: POKE
     34,8: HOME
520  PRINT "PRESS A KEY TO READ WITH MOTOR CONTROL";: CALL  - 756
     : PRINT : PRINT
530 :
540 :
550  REM  READ WITH MOTOR CONTROL
560 :
570  REM  TAKES ONLY 41.5 SECONDS WITH MOTOR PATCH ACTIVE
580 :
590  REM   IMPORTANT TO USE 'ON ERR GOTO' !
600 :
610  ONERR  GOTO 930
620 :
630  GOSUB 820: REM  SET MOTOR CONTROL PATCH
640 :
650  PRINT D$;"OPEN TEST"
660  PRINT D$;"READ TEST"
670  FOR N = 1 TO 100
680  INPUT "";T$
690 :
700  REM  SIMULATE PROCESSING INPUT DATA
710  FOR C = 1 TO D: NEXT C
720 :
730  NEXT N
740  PRINT D$;"CLOSE"
750 :
760  PRINT : PRINT "IMPROVED READ COMPLETE": PRINT  CHR$ (7): PRINT

770  PRINT D$;"DELETE TEST"
780  POKE 216,0: REM  DON'T NEED ONERR GOTO NOW
790  GOSUB 870: REM  RESTORE NORMAL DOS
800  HOME : PRINT "TEST COMPLETE....": TEXT : END
810 :
820  REM  SET MOTOR PATCH HERE
830 :
840  IF  PEEK ( - 16834) = 189 THEN  POKE  - 16834,96: REM  DOS 3
     .2.1 (CHANGE -16834 TO -16819 FOR DOS 3.3)
850  RETURN
860 :
870  REM  RESTORE MOTOR PATCH HERE
880 :
890  IF  PEEK ( - 16834) = 96 THEN  POKE  - 16834,189: REM  RESTO
     RE DOS 3.2.1 ORIGIONAL VALUE (CHANGE -16834 TO -16819 FOR DO
     S 3.3)
900  POKE  - 16152,0: REM  TURN MOTOR OFF
910  RETURN
920 :
930  REM  ON ERR ROUTINE
940 :
950  GOSUB 870: REM  SHOULD RESTORE ORIGIONAL DOS VALUES AND STOP
      MOTOR
960  POKE 216,0: HOME : VTAB 5: PRINT "DISK ERROR": PRINT : PRINT
     "PRESS ANY KEY TO TRY AGAIN.. ";: CALL  - 756: PRINT : GOTO
     550
970  HTAB 20 -  LEN (O$) / 2: PRINT O$: RETURN : REM  CENTERING R
     OUTINE
```
©

Interfacing The Apple To 6500 Family Peripherals

David Paul and James Wisman
Department of Chemistry
University of Arkansas
Fayetteville, AR

It has been stated previously that 6500 peripheral chips (6522, 6551, etc.) accessed from the Apple II peripheral bus must undergo complete address decoding. This results from the fact that 6500 chips require stable address and chip select lines 180 nanoseconds before the positive edge of the Φ2 clock (1). In this paper, a simple delay circuit allows use of the Apple device select lines to avoid the complexity of full address decoding.

Initially, a test program was written to explore the timing between Φ0 and device select on the Apple II peripheral bus. The test program was as follows:

```
0300 AD   B0   C0   over   LDA slot address
0303 4C   00   03          JMP over
```

This simple program produces continuous low going pulses on the address device select line. Observation via dual channel oscilloscope showed the relationship between Apple peripheral bus lines, device select, and Φ0. Figure 1 illustrates why 6500 family chips have difficulty working with the Apple II peripheral bus using the device select lines. Note that the falling edge of device select corresponds with the rising edge of Φ0. The device select line represents the address decode of the high order address lines that have been logically ANDed with Φ0. For the 6500 family, address lines must be stable 180 nanoseconds before the positive transition of the Φ2 clock line (2). (The Φ2 of 6522 is Φ0 from the Apple II Peripheral Bus.) Figure 1 shows absence of the needed delay. The solution calls for development of circuitry to delay the positive edge of Φ0, making the device select line useful.

A positive edge delay circuit is shown in Figure 2, with a corresponding timing diagram, Figure 3. In the resulting output waveform, the positive edge has been delayed several nanoseconds. The basic concept of the delay circuit centers around the voltage required to trigger a TTL gate. For the 74LS08 AND gate, potentials greater than a threshold of 1.0 volts are considered to be logical "highs." Rationalization for the delay circuit is as follows. When the input goes low, the output of the first AND gate (Pin 3) goes low: the diode is forward biased, discharging the capacitor so that both inputs to the second AND gate are low, resulting in a low output. When the input rises high, the output of the first AND gate goes high to reverse bias the diode. Current is then in the RC network charging the capacitor. The voltage across the capacitor rises until the 1.0V threshold level is reached. At this potential, the second AND gate recognizes the voltage level as a high (Pin 5), changing the state of the output.

The magnitude of the delay can be adjusted by changing the value of the RC time constant. The voltage drop across the capacitor (V_{TTL}) can be expressed by:

$$V_{TTL} = V(1-e^{-t/RC})$$

where: V - the TTL high (4.0V)
t - time
RC - resistor and capacitor values.

Rearranging, this equation takes the form:

$$t_{delay} = \left[-R \ln\left(1-\frac{V_{TTL}}{V}\right)\right]C$$

where: t_{delay} - the rising edge delay
V_{TTL} - threshold voltage 1.0V

Therefore, for a constant resistance R, the amount of delay is proportional to the size of the capacitor (3). For example, for R=1.5kΩ and C=330pf, the predicted delay would be 140 nanoseconds, which agrees with that obtained experimentally. The upper limit of the delay results from an RC time constant that will not allow the capacitor to reach the 1.0V threshold during the 1/2 cycle time of ~500 nanoseconds. For R=.5kΩ, the maximum capacitor size is about 2000pf.

Advantages of this circuit are: 1) a TTL level signal results, 2) the falling or trailing edge is unaffected, and 3) inexpensive and simple to incorporate into existing peripheral design.

Implementation of delay circuit utilizing a 6500 family peripheral is shown in Figure 4. A 6522 VIA was chosen to test the applicability of the delay circuit. A simple inspection of memory through the Apple monitor can determine if the 6522 registers are communicating with the Apple. Simple software routines to define data direction, to read and write data to both 8 bit ports, and to set and decrement timers were used to determine that the 6522 was fully functional when using the delay circuit. All features of the 6522 are non-functional with no positive edge delay of Φ0.

Even though all 6500 family chips have not been tested, the authors presume that this delay technique is applicable to all 65XX peripheral devices. This investigation has revealed one excep-

HUNTINGTON COMPUTING

PROGRAMS LISTED BELOW ARE ON DISK:

Sabotage	$24.95	now	$21.19
HI-RES Soccer	$29.95	now	$25.39
Pulsar II	$29.95	now	$25.39
Orbitron	$29.95	now	$25.39
Autobahn	$29.95	now	$25.39
Gamma Goblins	$29.95	now	$25.39
EPYX (Automated Simulations) SPECIAL — While they last: Ryn/Morloc/Rigel all three			$39.99
Galaxy Wars	$24.95	now	$22.99
Super Scribe	$129.95	now	$110.39
Galactic Empire	$24.95	now	$21.20
Galactic Trader	$24.95	now	$21.20
Galactic Revolution	$24.95	now	$21.20
DR Master	$229.00	now	$194.59
Aristotle's Apple	$34.99	now	$29.99
Micro Memo	$39.95	now	$35.99
The Data Factory	$150.00	now	$124.00
Mission Asteroid	$19.95	now	$17.99
Galactic Attack	$29.95	now	$25.39
Wizardry	$39.95	now	$33.99
Bright Pen	$49.95	now	$42.49
Star Mines	$29.95	now	$25.39
PFS (Personal Filing System)	$95.00	now	$80.69
PFS: Report	$95.00	now	$80.69
Warp Factor	$39.95	now	$33.99
Vinyl holders for 20 disks in beautiful deluxe padded binder			$13.99
DOS Tool Kit	$75.00	now	$69.00
Microsoft Adventure	$29.95	now	$26.99
Olympic Decathlon	$24.95	now	$21.99
Compu-Math: Arithmetic	$49.95	now	$44.95
CRAE 2.0	$24.95	now	$21.99
MCAT 2.0	$19.95	now	$17.99
Cartels and Cutthroats	$39.95	now	$33.99
Operation Apocalypse	$59.95	now	$50.99
Torpedo Fire	$59.95	now	$50.99
Tank Attack	$20.00	now	$16.99
Modifiable Database II	$150.00	now	$127.49
TG Game Paddles	$39.95	now	$33.99
TG Joy Stick	$59.95	now	$50.99
The Tarturian	$24.95	now	$21.99
The Prisoner	$29.95	now	$26.99
The Wizard & The Princess	$32.95	now	$28.99
Flight Simulator (disk)	$34.95	now	$29.50
Star Cruiser	$24.95	now	$20.99
Space Album	$39.95	now	$34.50
Odyssey	$29.95	now	$25.50
Micro League Baseball	$24.95	now	$22.99
Sargon II	$34.95	now	$29.70
Program Line Editor	$40.00	now	$34.99
Space Eggs	$29.95	now	$25.39
Mystery House in French			$22.99
Z-80 Softcard	$349.00	now	$299.00
Videx 80-Col. Board	$350.00	now	$299.00
3D Super Graphics	$39.95	now	$33.99
Compu-Math I or II	$40.00	now	$34 ea.
HI-RES Cribbage	$24.95	now	$21.99
Phantom Five	$29.95	now	$25.49
Star Gazer's Guide	$30.00	now	$25.39
Lords of Karma	$20.00	now	$16.99
Apple PIE & Formatter (Reg. $129.95)		Special	$99.99
The Book of Software	$19.95	now	$17.99
Versa Writer Expansion Pac-I	$39.95	now	$33.99
The Linguist	$40.00	now	$33.99
Apple II Users Guide	$14.95	now	$12.99
Oh Shoot!			$19.99
Serendipity Statistics or Gradebook	$169.00	now	$143.59
Bridge Partner (disk)	$24.95	now	$21.19
ABM (Muse)	$24.95	now	$22.49
Data Plot (Muse)	$59.95	now	$52.99
Computer Conflict	$39.95	now	$33.99
Computer Air Combat	$59.95	now	$52.99
The Temple of Apshai	$39.95	now	$33.95
Super-Text II	$150.00	now	$128.99
Magic Window	$99.95	now	$85.00
Request	$225.00	now	$191.19
Space Raiders	$29.95	now	$25.39
Thinker	$495.00	now	$420.69
Super Kram	$175.00	now	$148.69
Savage Island I (disk)	$20.95	now	$17.79
Savage Island II (disk)	$20.95	now	$17.79
Thrilogy	$24.95	now	$21.19
Zork	$39.95	now	$33.99
Howardsoft Real Estate Analyzer	$150.00	now	$129.99
Super Disk Copy-Sensible	$30.00	now	$25.49
Fastgammon	$24.95	now	$21.20
Astro Apple	$20.00	now	$17.45
Akalabeth	$34.95	now	$29.99

Use Our new 800 Number (except Calif.) 800-344-4111

Use Our new 800 Number (except Calif.) 800-344-4111

MEMOREX DISKS 10 for $24.99

VISA

master charge

We take MasterCard or VISA (include card # and expiration date). California residents add 6% tax. Include $2.00 for postage. Foreign and hardware extra. Send for free catalog. Prices subject to change.

HUNTINGTON COMPUTING, **SUPER DISCOUNTS**
2020 Charles Street
Post Office Box 787
Corcoran, California 93212

apple® SOFTWARE

Order by Phone (209) 992-5411

APPLE II*
SOFTWARE FROM
POWERSOFT

PEGASUS

(a PASCAL based data base system)

PEGASUS – is a filing and retrieval system using the PASCAL programming language providing a general means for storing data in an orderly fashion. PASCAL code runs three to five times faster than BASIC code designed for a similar application.

Data stored in the PEGASUS data base may be modified, retrieved, and formatted into convenient reports. Three types of data are supported: character, real, and integer. Each PEGASUS data base record may contain up to 20 fields.

Data may be entered either interactively from the console or as a batch from a text file. Records may be modified after they have been entered or deleted from the data base entirely. PEGASUS may also be used to select groups of records based on the values of one or more fields. Output may be to the CRT screen, a printer or a text file. Thus, PEGASUS may be used to create printed reports, examine data on-line, or *interface with the input or output of other PASCAL programs.* Requirements: Apple II, Plus, or III and two 5¼" disk drives. Or an 8" or Winchester type drive. USCD Pascal Language System. . . . .
5¼" Disk Only/$199.95

INCOME STATEMENT SYSTEM

INCOME STATEMENT SYSTEM – (Summarized Reports including Budget Figures Based on Super Checkbook III transactions.) – An excellent program complement to SUPER CHECKBOOK III. The system provides for up to 100 income and expense codes. For each code the system maintains a total for the current month, current budget, current year-to-date, and three prior year-to-dates. Income codes may have up to six corresponding expense codes. A "sort code" feature allows account codes to print in a user defined sequence.

Updates to the accounts include current month, end-of-month, and end-of-year. Gross and Net Income Statements may be printed in either account code or sort code sequence. The Account Master File List may be printed by sort code, account code, or alphabetically by account name. Detailed transactions for each code are printed and totaled, one code per page, in code number order.

This system is designed to run in conjunction with the SUPER CHECKBOOK III program described below. Requirements: 48K, two disk drives, printer card, Applesoft
Disk Only/$49.95

SUPER CHECKBOOK III

SUPER CHECKBOOK III – A *vastly improved* version of our popular selling program. With new features such as: simplified but powerful transaction entry and modification routines, new reconciliation routines, additional features such as 30 percent increase in the total number of checks handled, posting of interest from interest bearing checking accounts, automatic teller transactions, bullet proof error handling, and smart disk routines. Plus the program still contains the options of bar graphs, sorting, activities, and account status. See INCOME STATEMENT SYSTEM described above.
Disk Only/Applesoft $49.95

Dealer Inquiries Invited

Visa and MasterCard, Check or Money Order include $2.00 for shipping and handling. C.O.D. $2.00 additional.

*Apple II and Applesoft are registered trademarks of APPLE COMPUTER INC.

POWERSOFT
P. O. BOX 157
PITMAN, NEW JERSEY 08071
(609) 589-5500

tion to the Φ0 delay requirement. The 6520 PIA will function properly when connected directly to the Apple Φ0 and device select lines.

In summary, the delay of the positive edge of Φ0 from the Apple peripheral bus results in simpler interfacing with the 6500 family. Eliminating full address decoding utilizes slot independence of peripheral cards that are less expensive and easier to construct.

References

1. De Jong, Marvin C. **COMPUTE!**, 3, #3. pg. 142
2. Rockwell Data Sheet Document NO: 29000 D47, Revision, 2, June 1979.
3. Diefenderfer, A. James; Principles of Electronic Instrumentation, 2nd Ed. W. B. Sanders; Co. 1979.

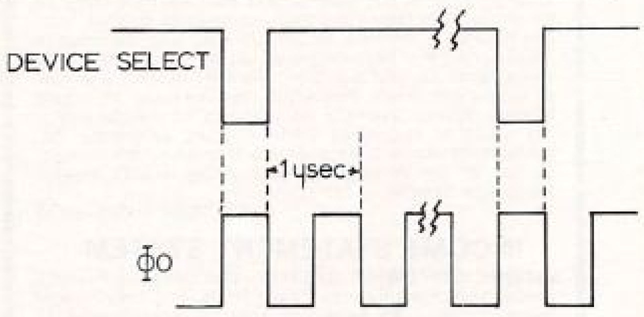


Figure 1.

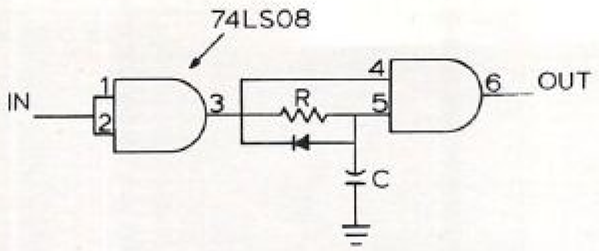


Figure 2.

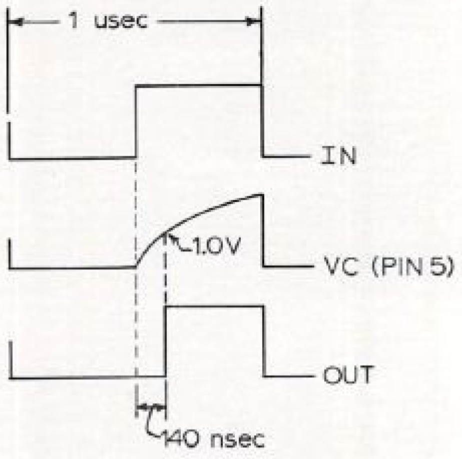


Figure 3.

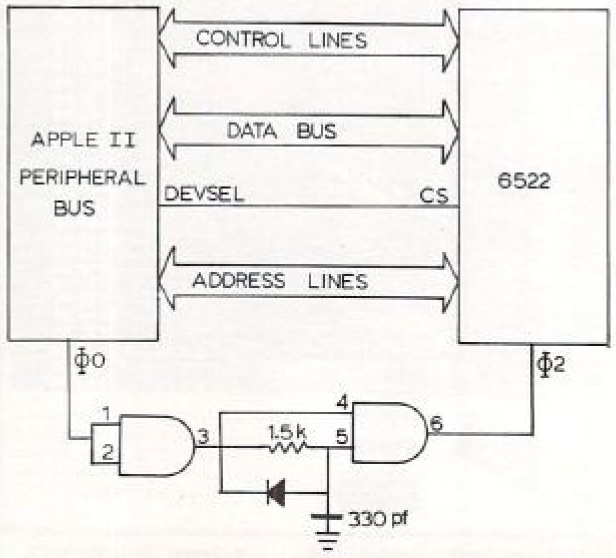
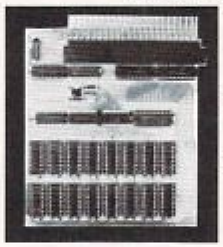


Figure 4.

AIM 65-8K STATIC MEMORY



- ★ Plugs directly onto AIM-65 memory expansion blade.
- ★ Positions neatly under AIM-65 allowing use of available enclosures.
- ★ Expansion blade provided for further expansion.
- ★ +5 volts supplied by host AIM-65.
- ★ 8K memory board draws only 200 ma.
- ★ Two separately addressable 4K blocks.
- ★ KIM-1 compatible.

MEM 4:	8K memory board, 4K RAM chips	$109.00
MEM 8:	8K memory board, 8K RAM chips	$169.00
RAM 4:	4K RAM chips to upgrade MEM 4 to 8K	$ 69.00
Full documentation kit		$ 1.00

SYSTEM PERIPHERALS
P.O. Box 971, Dept. C.
Troy, MI 48099

ELEMENTARY SCHOOL MATH PROGRAMS

Apple II or III 32K and Up.

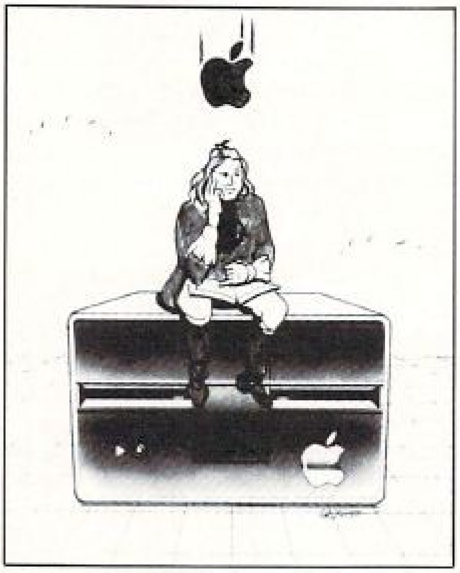
Volume I: Mult. Tables — Primary Math
Volume II: Middle & Adv. Elem. Math
Volume III: Fractions, Decimals, and Percents. Grade levels 1st thru 9th

All programs are add, sub., mult. & div. All are corrected, graded, and have both visual and auditory rewards.

Disks: $15./volume; All 3 $40.00.
Cassettes: $10.00; All 3 $25.00

PIPES COMPUTER SOFTWARE
ROUTE 1, BOX 284
MURPHY, NC 28906
Send Check, Money order or P.O.

Beneath Apple DOS

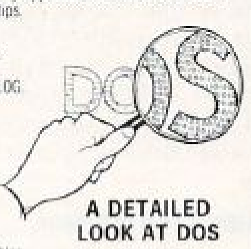


A technical manual by Don Worth and Pieter Lechner

Become an expert on the intricacies of Apple's DOS (Disk Operating System). BENEATH APPLE DOS is the perfect companion to Apple's DOS 3.3 Manual. Containing eight chapters, three appendices, a glossary, an index, and over 160 pages, this manual will serve to completely fill in the many gaps left by Apple's DOS 3.3 Manual. Written for Apple users with DOS 3.3, 3.2 or earlier versions, any Apple disk user would welcome having this carefully written manual at his fingertips.

LEARN . . .

- How DOS 3.3 differs from other DOS versions.
- How disks are protected.
- How to reconstruct a damaged diskette CATALOG.
- How tracks are formatted.
- How to use the disk directly, without DOS.
- How to call DOS's file manager.
- How *every* routine in DOS works.
- How to customize DOS to your needs.
- How to overcome DISK I/O ERRORS.
- About the "secret" file types — S and R.



A DETAILED LOOK AT DOS

INCLUDES . . .

- Large quantities of excellent diagrams and tables.
- Source listings of useful disk utilities.
- Glossary of over 150 technical terms.
- Exhaustive description of DOS program logic.
- Handy reference card.
- Useful patches to DOS.
- Many programming examples.

Book - $19.95

CALL OR WRITE FOR OUR CATALOG OF SOFTWARE PROGRAMS FOR THE APPLE II AND APPLE II PLUS*

WHERE TO GET IT: Call us for the name of the Quality Software dealer nearest you. If necessary you may order directly from us. Mastercard and Visa cardholders may place orders by telephone. Or mail in your check or bankcard number. California residents add 6% sales tax. SHIPPING CHARGES: Within North America orders must include $1.50 for first class shipping and handling. Outside North America the charge for airmail shipping and handling is $5.00. Pay in U.S. currency.

QS QUALITY SOFTWARE
6660 Reseda Blvd., Suite 105, Reseda, CA 91335
(213) 344-6599

*Apple II and Apple II Plus are trademarks of Apple Computer, Inc.

A Cassette Tape Monitor For The Apple

Jim Lowell
Whitehouse Station, NJ

"Type LOAD; do not press RETURN yet. Remove the plug from the earphone jack of the tape recorder. Press the PLAY button on the recorder and advance the tape until you hear the leader tone. Stop the recorder and replace the plug in the earphone jack. Start the tape, and immediately press RETURN."

The Tape Monitor

To most tape-recorder-based APPLE computer users, the complex instructions above are a source of continued annoyance and frustration. Fortunately, they are also unnecessary.

The tape monitor unit described in this article eliminates the need to remove the plug from the earphone jack and simplifies the tape loading instructions to: "Type LOAD; do not press RETURN yet. Press the PLAY button on the tape recorder. When you hear the leader tone, press RETURN."

For about one hour's work and one-half the cost of a good tape program (about $12.00), you can build a tape monitor unit for your APPLE. To simplify the construction process, I have included a set of templates to locate the required mounting holes and both a schematic diagram (for those of you who are electrically inclined) and a wiring diagram (for those of you who aren't).

Using The Monitor

Use of the monitor unit has several advantages over the remove-the-plug method of tape loading:

- It simplifies the loading procedure.
- It allows the user to listen to the tape as it is loaded.
- It eliminates the need to change the tape recorder volume control setting to avoid disturbing others while listening to the leader tone.
- It reduces the wear and tear on the earphone jack of the recorder and the mini-plug of the computer connecting cable.

To use the monitor for the first time, connect the cord from the *tape in* jack of the APPLE to the mini-jack of the monitor. Then, insert the mini-plug of the monitor into the earphone jack of the tape recorder. Put a tape program into the machine and press PLAY. Adjust the volume control on the monitor (don't change the setting on the recorder yet) until it is at the desired level. Now, rewind the tape and go through the following loading steps:

- Type LOAD without hitting return.
- Start the tape.
- When you hear the leader tone, hit RETURN.
- When the cursor re-appears, rewind the tape.

If the tape fails to load properly, the problem could be that the speaker in the monitor is drawing some of the required signal away from the computer (this has never been a problem with my Panasonic recorder — I load most tapes at a volume setting of "4"). If the problem occurs, however, turn the recorder volume up a notch or two — that should fix it.

Building The Monitor

The parts list below uses mostly Radio Shack components because they are widely available; any comparable parts will do.

Here are my suggested steps for building the monitor.

1. Using the templates, locate and drill all holes in both the top and case.

2. Fasten all the proper parts to the top of the project case (i.e. everything but the mini-jack, mini-plug, control knob, and cord). Be sure you mount the speaker cloth over the right hole before mounting the speaker.

3. Connect these parts together as per the schematic or wiring diagrams (using single-conductor copper wire). The wires coming from the switch and volume control to the mini-jack should be about eight inches long. This is to allow the top of the project case to be easily removed if necessary. Don't connect the wires to the mini-jack yet.

4. Put the rubber grommet in the proper hole in the side of the project case (the one where the cord will go).

5. Connect the mini-plug to about one foot of shielded cable (the shield goes to the "ring" terminal of the plug), and thread the other end of the cable (with the shield and conductor already stripped and tinned) through the grommet into the case.

6. Twist the shield of the cord and the end of the wire from the volume control together, and solder them to the "ring" terminal of the mini-jack.

7. Twist the conductor of the cord and the end of the wire coming from the switch together and solder them to the "tip" terminal of the mini-jack.

8. Fasten the mini-jack in position in its hole in the side of the case.

9. Screw on the top of the case, put the control knob onto the volume control post and you're done.

I've tried to supply all the necessary instructions to make building the tape monitor an easy evening's project. If, however, you encounter any problems, drop me a line at the address below:

Jim Lowell
P.O. Box 364
Whitehouse Station, NJ 08889

PARTS LIST

COMPONENT	PART#
1. Project case (4-3/4" x 2-1/2" x 1-2/5")	270-222
2. 2" speaker (8 ohm)	40-245
3. Mini-switch	275-662
4. Mini-volume control (5K ohm)	271-214
5. Control knob	274-415
6. Mini-jack	274-296
7. Mini-plug (shielded)	274-288
8. 1½ feet of single-conductor shielded cable.	
9. 1½ feet of single conductor copper wire.	
10. 1½ inch piece of speaker cloth (or loose-weave fabric).	
11. A rubber grommet to fit a ½-inch hole.	

LOGICAL SOFTWARE, INC.
ANNOUNCES:

MAIL EXPRESS

A NEW MAIL LIST UTILITY FOR THE APPLE II.

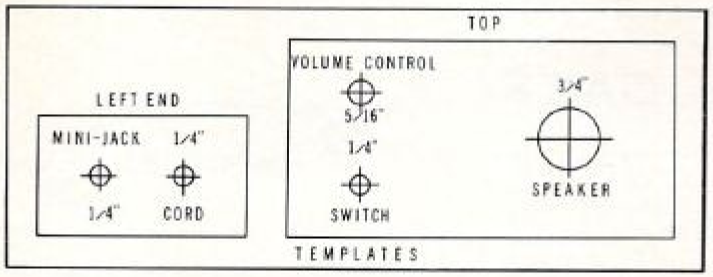
— Up to 2,200 Names per File
— Sort by Company Name, Customer Name, City, State Zip
— Prints Return Addresses
— Merge up to 16 Files
— Easy User Definable Codes for City, State and Zip to Save Time and Disk Space

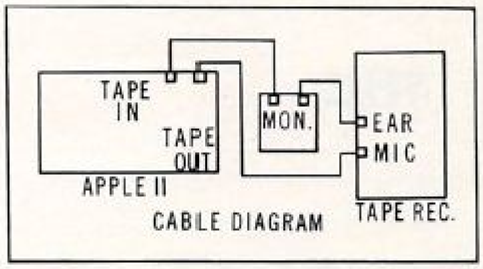
This is an easy to use professional quality mail list able to handle large or small files.

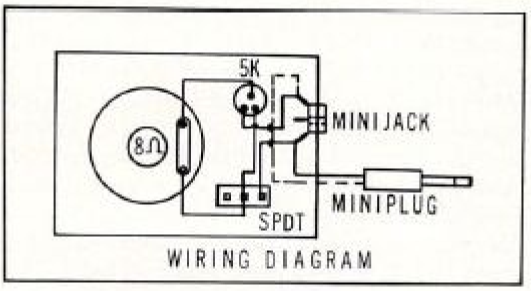
Introductory Price $49.95
$2.00 Postage & Handling

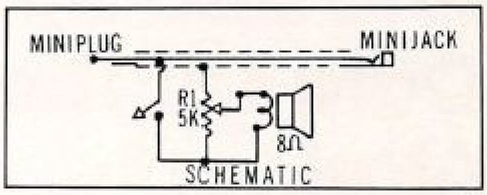
Logical Software, Inc.
P.O. Box 354
Farmington, MI 48024
(313) 474-8774

®Apple and Apple II are registered trademarks of Apple Computer Inc.









©

Diskette Sector Space In A Greeting Program

R. R. Hiatt
St. Catherines, Canada

Most Apple users include the system command CATALOG in their greeting program so that when the DOS is booted, the diskette's contents are automatically listed. CATALOG, of course, displays the number of sectors used for each file as well as the file name. What it does not report is the total number of used sectors, or the remaining number unused. To find out this rather useful information, the user must sum the individual numbers and subtract from 403. It's a small chore, but annoying, particularly as the computer should be doing the arithmetic, not the user.

There are two problems: The first, that the numbers are non-resident (except in the screen memory area) is easily solved by PEEKing them out. My program does this, and works quite well as long as the diskette holds fewer than 24 files. It will always sum and report for the last 23 files, but that can be misleading. Consequently, I restrict myself to 23 files per diskette.

The real problem lies in the construction of the CATALOG routine itself. The output is paged, but control is not returned to the program until the entire contents have been screened. Thus, the program stands by helplessly as the numbers it's supposed to PEEK scroll upwards, off the screen, and into oblivion.

It would be an easy matter to amend CATALOG if one knew where it was. But searching the 12K bytes of disassembled DOS 3.2 in the hope of recognizing the routine seems more work than it's worth. Until some thoughtful person publishes a DOS map, I'll be content with my present, admittedly imperfect, but useful, program.

```
10  REM GREETING W CATALOG & ADD
20  REM SECTOR ADD FOR 23 FILES OR FEWER
30  REM CATALOG MUST FINISH BEFORE ADD
WILL START
40  PRINT "SLAVE DISK INIT ON 48K APPLE"
50  PRINT "BY R. HIATT, 12/1/80"
60 D$ = CHR$(4)
70 S = 128:D = S + 48
80  FOR I = 1 TO 2000: NEXT I: HOME
90  PRINT D$;"CATALOG"
100 T = 0:CT = 0
110  FOR BASE = 1024 TO 1104 STEP 40
120  FOR J = 0 TO 7
130 ROW = BASE + J * S
140 T1 = T
150  FOR COL = 3 TO 5
160 A = PEEK (ROW + COL) - D
170  IF A < 0 OR A > 9 THEN 190
180 T = T + A * 10 ↑ (5 - COL)
190  NEXT COL: IF T > T1 THEN CT = CT + 1
200  NEXT J: NEXT BASE
210  PRINT : PRINT "SECTORS USED TOTAL
";T
220  PRINT : PRINT "UNUSED SECTORS =
";403-T
230  PRINT "NUMBER OF FILES = ";CT
240 END
```

©

Mountain Computer CPS MultiFunction Card™

The Only Interface Card You Need!

Connecting a Parallel Printer?

Epson®- Centronics®- IDS Paper Tiger®- CPS handles all these printers and others with on-board intelligence to provide paging and other features found on no other card.

Connecting a Serial Printer?

Diablo®- Qume®- NEC - TI 800 Series®- CPS handles these printers and others with standard RS-232 interface providing selection of baud rates, handshakes, paging, and more.

Connecting a Modem or Terminal?

Hayes Smartmodem®- Novation CAT®- M & R Pennywhistle®- CPS handles these and others with full/half duplex operation, baud rate selection, and even a transparent terminal mode which includes a dual mode feature that permits printing of text to parallel printers while 'on-line' eliminating the need for special terminal software—and more.

Connecting with the Time?

The on-board calendar/clock provides real time and date information including day of week, day, month, year, hours, minutes, and seconds for any application requiring a time stamp—battery backed-up for over two years!

Connecting with The Source?

Used with a modem, CPS provides the connection to information utilities, such as The Source,® Dow Jones, and others. Additionally, CPS provides the connection to big-time electronic mail with programs such as Micro Courier® and Micro Telegram® and other data transfer programs.

Connect with Easy Use!

CPS has no switches to set! All functions on the card are set from a user program. Menu driven screens set up your choice of all functions on CPS and store them on-board in CMOS RAM—battery backed-up (including the clock) for over two years! To change parameters, run the set-up program again—or use special commands from your keyboard. Furthermore, most existing software programs are immediately usable with CPS. *Phantom Slot Capability* permits assignment of CPS' functions to your software's pre-defined slots.

Connect with Your Apple® Dealer

Drop by your Apple dealer and see how the CPS MultiFunction Card provides the most comprehensive capabilities for RS-232C serial interface, parallel output, and real-time calendar/clock of any card available today—*all on one card*—at one low price—competitive with *any one* of the three or more single function cards that it replaces.

Mountain Computer
INCORPORATED
300 El Pueblo Road, Scotts Valley, CA 95066
(408) 438-6650 TWX: 910 598-4504

® Registered trademarks

Turn your Apple into the world's most versatile personal computer.

The SoftCard™ Solution. SoftCard turns your Apple into two computers. A Z-80 and a 6502. By adding a Z-80 microprocessor and CP/M to your Apple, SoftCard turns your Apple into a CP/M based machine. That means you can access the single largest body of microcomputer software in existence. Two computers in one. And, the advantages of both.

Plug and go. The SoftCard system starts with a Z-80 based circuit card. Just plug it into any slot (except 0) of your Apple. No modifications required. SoftCard supports most of your Apple peripherals, and, in 6502-mode, your Apple is still your Apple.

CP/M for your Apple. You get CP/M on disk with the SoftCard package. It's a powerful and simple-to-use operating system. It supports more software than any other microcomputer operating system. And that's the key to the versatility of the SoftCard/Apple.

BASIC included. A powerful tool, BASIC-80 is included in the SoftCard package. Running under CP/M, ANSI Standard BASIC-80 is the most powerful microcomputer BASIC available. It includes extensive disk I/O statements, error trapping, integer variables, 16-digit precision, extensive EDIT commands and string functions, high and low-res Apple graphics, PRINT USING, CHAIN and COMMON, plus many additional commands. And, it's a BASIC you can compile with Microsoft's BASIC Compiler.

More languages. With SoftCard and CP/M, you can add Microsoft's ANSI Standard COBOL, and FORTRAN, or Basic Compiler and Assembly Language Development System. All, more powerful tools for your Apple.

Seeing is believing. See the SoftCard in operation at your Microsoft or Apple dealer. We think you'll agree that the SoftCard turns your Apple into the world's most versatile personal computer.

Complete information? It's at your dealer's now. Or, we'll send it to you and include a dealer list. Write us. Call us.

SoftCard is a trademark of Microsoft. Apple II and Apple II Plus are registered trademarks of Apple Computer. Z-80 is a registered trademark of Zilog, Inc. CP/M is a registered trademark of Digital Research, Inc.

MICROSOFT
CONSUMER PRODUCTS

Microsoft Consumer Products, 400 108th Ave. N.E., Bellevue, WA 98004. (206) 454-1315

A Tape "EXEC" For Applesoft: Loading Machine Language Programs

Sherm Ostrowsky
Goleta, CA

Apple owners with Disk systems have available a very powerful DOS command, "EXEC", which will effectively turn control of the computer over to a text file on the disk. The lines in this file are treated as if they had been typed in at the keyboard in Immediate mode, and are executed. Unfortunately, we owners of "obsolete" cassette-tape based systems don't have the benefit of this capability. But, in this article I will show you how to obtain some of the power of an "EXEC" file on tape. I'll demonstrate the method, which is actually quite general, by showing how to load Machine-Language (ML) programs just as easily as you now load Applesoft programs, and how to combine Applesoft and ML loads on one cassette in an effective manner. It has been said that most apple owners have disks, but I suspect that those who still use tape include a high proportion of relative beginners, so this article will be slanted toward them.

Some of the programs in my library are in Applesoft and others are in ML, but all of them are still on cassettes. As you are probably aware, these two different types of programs must be loaded into the computer by entirely different commands. An Applesoft program is loaded very simply, by typing LOAD. You don't have to know how long the program is or where in memory it is supposed to be stored; Applesoft takes care of all those details for you. But a ML program is a pain in the neck to load. First you have to enter the Monitor by typing CALL -151. Then you have to know the exact addresses of the beginning and end of the program, so you can type:

(Begin Address).(End Address)R

to start the loading process. And woe unto you if you are off by even one byte in remembering where the programm is supposed to go: you'll get that dreaded "beep" and "ERR" message.

And, after it's loaded, the difference between Applesoft and ML programs continues to exist, to the discomfort of the latter. To run the Applesoft program, you type RUN — what could be simpler? To run the ML program you have to know its Entry Address, which may or may not be the same as its Begin Address; then you type (still in the Monitor)

(Entry Address)G

to get it started. You have to keep referring to written notes in order to load and run a ML program successfully.

Well, I got tired of all this. I wanted to load all my programs, whether in Applesoft or ML, in exactly the same way — by typing LOAD. And I wanted to run them all the same way — by typing RUN. The computer has a better memory than I have, so let *it* keep track of where the darn ML program begins and ends, and where to enter it. After a while, I found a way to do this, and I'll describe it to you below. In the process, I discovered that the method would also solve some other problems connected with how to combine Applesoft Programs with ML subroutines in a convenient fashion. These, too, I shall pass on to you.

Although the method I am about to describe is very easy to use, it is actually based upon some rather intimate details concerning the inner workings of Applesoft. So, as a byproduct, I hope this article will add to your knowledge in this area, so vital to making fullest use of the capabilities of the Apple.

Let us begin by solving the problem of how to simplify the loading and running of a single ML program. We'll assume that you start out with the program already in the computer's memory, having been loaded (for the very last time, let us hope) by the tedious old method. We must also assume that you can, if you wish, SAVE the ML program back onto a cassette tape by typing:

(Begin Address).(End Address)W

in the Monitor. This last assumption may be more of a stumbling block than you may think, since some commercial programs are "protected" so that they cannot easily be copied, i.e., SAVEd onto another cassette. Sorry, folks, but if that is the case with your program, then I can't help you.

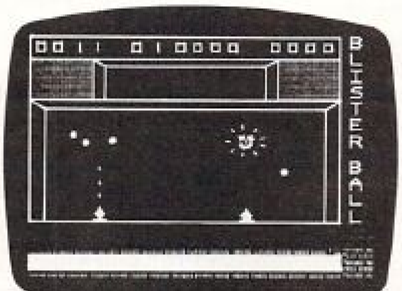
Now, leave the Monitor and enter the Applesoft level by typing Control-C (Return), and type in an Applesoft loader program like the one I'm going to show you below. The example is for a

The BEST games are from Creative Computing Software

1978: Adventure
1979: Air Traffic Controller
1980: Super Invader
1981: Blister Ball and Mad Bomber

Blister Ball

Blister Ball is the first completely original arcade-type game for a computer. Not a copy, not an adaptation, not a spinoff. **Blister Ball** is new—it's a new idea—better than Invaders, better than Circus, better than Asteroids, better than Galaxian. If you've played other games for hours, you'll play **Blister Ball** for days.



How does it work? Well, some mean but fun-loving aliens have produced some bouncing bombs. First they drop one and you've got to position yourself under it and zap it with your laser. If you miss, that's OK. It will bounce around, although each bounce is lower, and you have several chances to zap it. Got the hang of it? OK, here come two bouncing bombs. You zap them. Then you're faced with three, then four and five.

As they bounce longer and longer the walls begin to close in so you're faced with either zapping the bombs or being hit. Each hit knocks you a little further toward the gutter. But you can survive two hits which is usually enough to zap all the bombs.

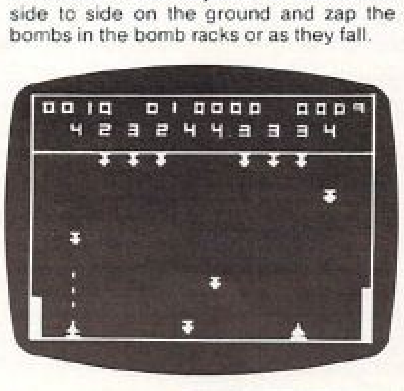
Feeling confident? Don't. Because after 5 bombs the murderous little devils drop 5 bonus bombs, worth ten times as much. These don't bounce, so you get only one shot. You need nerves of steel and the reflexes of a tail gunner.

After you complete one round, the game starts again with bombs that bounce faster and lower (and are worth more) than the previous ones.

Blister Ball is a fantastic solo game. But there are two-player options as well in which players can play as a team or as opponents. Each player can move the entire width of the screen and zap any of the bombs. Here, you're not only trying to survive, but trying to outscore your opponent. The game has two skill levels.

Mad Bomber

In **Mad Bomber** you are faced with aliens in a huge ship hovering overhead. They have bomb racks which they constantly fill with bombs. Your object is to move from side to side on the ground and zap the bombs in the bomb racks or as they fall.



As the game progresses, the aliens fill up their bomb racks more quickly and the bombs fall faster. You lose after ten bombs have hit the area which you are defending.

Mad Bomber can be played by one player solo or by two players as a team or as opponents. Two skill levels.

Order Today

Blister Ball and **Mad Bomber** are available together for $29.95 on disk (DOS 3.2) only and require a 48K Apple with paddle controls. (We recommend using the Super Paddles from Peripherals Plus).

To order send $29.95 plus $2.00 shipping and handling to the address below. Credit card customers should include card number and expiration date of Visa, MasterCard or American Express card. Credit card orders may also be called in to our toll-free number in the continental U.S.

If you also wish to order a set of Super Paddles from our Peripherals Plus subsidiary, the cost is just $39.95. The paddles are backed by a 90-day limited warranty from the manufacturer as well as Peripherals Plus' moneyback guarantee of satisfaction.

Blister Ball and **Mad Bomber** are colorful, challenging, fast and noisy. They are the games of the year from Sensational Software.

creative computing

Attn: Hope
39 E. Hanover Avenue
Morris Plains, NJ 07950
Toll-free **800-631-8112**
In NJ 201-540-0445

specific program that I use a lot: my Assembler. And the example has had a few unnecessary bells and whistles added to it to enhance its convenience to me; you may want to leave these off for your application. Instead of describing the program in the usual way, with a lot of REM statements, I intend to do a far more thorough job of explaining it in the following text. So here's my Loader program, and the explainations come after it.

```
 10 REM APPLESOFT LOADER FOR
 20 REM THE S-C ASSEMBLER
 30 :
 40 HOME : VTAB 12: HTAB 8: PRINT
    "LOADING THE S-C ASSEMBLER"
 50 PRINT :X= POS(0)
 60 Y$ ="1000.24FFR   D823G"
 70 FOR I=1 TO LEN(Y$): POKE 511+I, ASC
    (MID$ (Y$,I,1) )) +128: NEXT
 80 POKE 72,0: CALL -144
 90 T=POS (0): IF T>X+1 THEN 200
100 POKE 214,85
110 PRINT CHR$ (7);"LOAD SUCCESSFUL —
    STOP TAPE": PRINT
120 FOR PAUSE =0 TO 2000: NEXT
130 CALL 4096: END
199 REM LOADING-ERROR EXIT
200 PRINT CHR$ (7); CHR$ (7); CHR$ (7);
    "*** LOADING ERROR ***": PRINT
210 END
```

Here is the explanation.

Lines 10–30 just tell what the program is for.

Line 40 is one of my "bells-and-whistles." It isn't necessary for proper operation of the program, but I find it comforting. It displays a message on the screen telling me what is going on, and keeps the message there for me to look at while the tape is being read. Some of these ML programs take a L-O-N-G time to load, and you sometimes begin to wonder if the computer is still doing anything.

Line 50 is also not strictly necessary, but it is very useful. It is part of an error detection scheme to keep me from trying to run the program if it didn't load in correctly. The Apple keeps a running tally of a checksum during the load process, and will give an "ERR" message if it fails to agree with the value that accompanies the program on the tape (thereby indicating that something has gone wrong in the loading), but, other than this message, the Apple doesn't set any error flags that can be read by a program. So here, before we even begin to load the tape, we record, in variable X, the horizontal position of the cursor. This will be used in line 90 (below) to determine if a loading error has taken place.

Lines 60–80 are the heart of the loader. They constitute a clever scheme by which an Applesoft program can, in effect, fool the computer into believing that you have typed in the line:

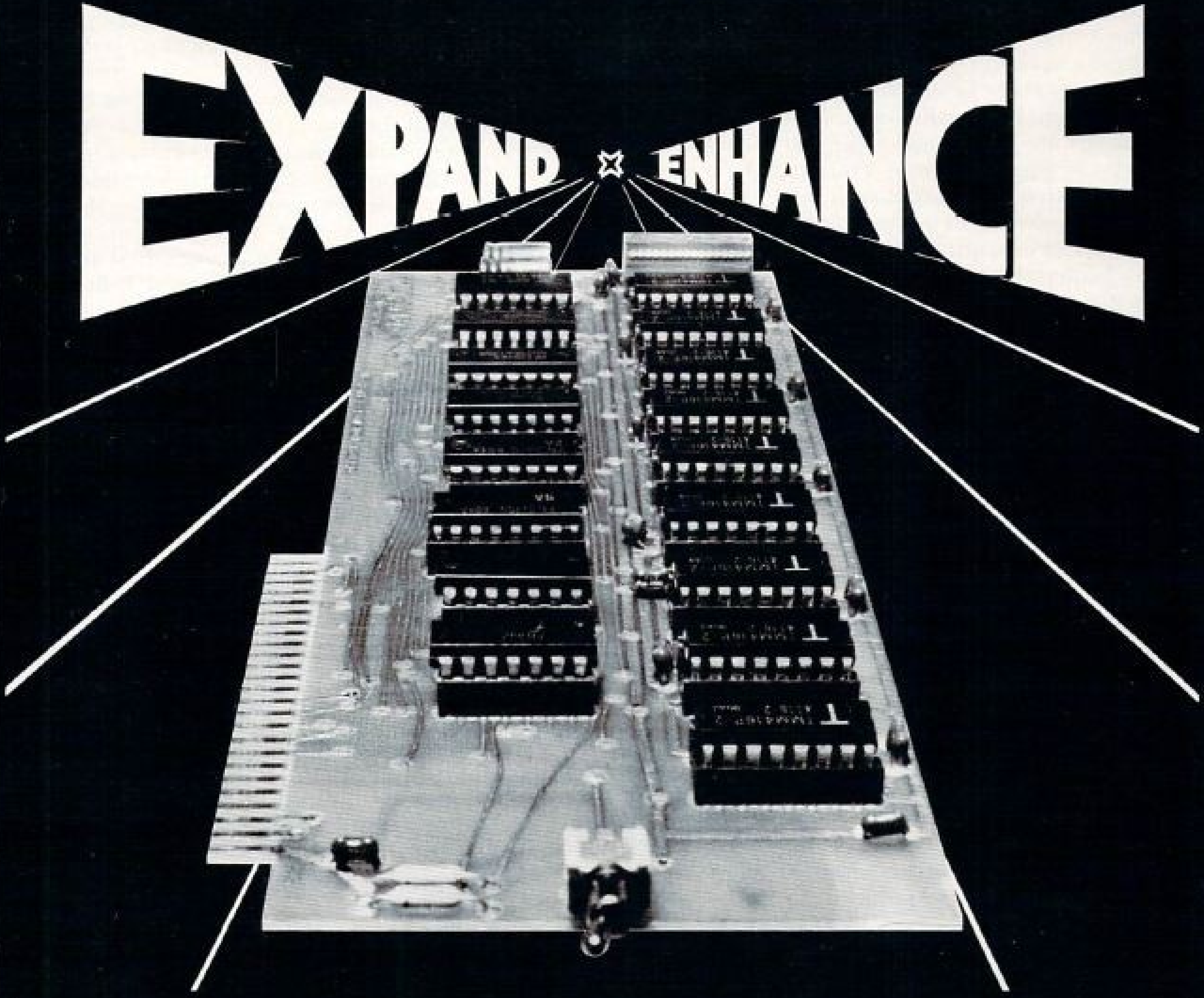
(Begin Address).(End Address)R

by way of the Monitor! It was invented by S.H. Lam. In line 60, the string variable Y$ contains a sequence of literal Monitor commands, just as you would have typed them in by way of the keyboard. The first part is the instruction to Load the ML program starting at address $1000 and ending at address $24FF (the "dollar sign" signifies a hexadecimal number, in 6502 notation). There follows an obligatory space, to separate this Monitor command from the next one. The second and last command on this line is "D823G", which tells the Monitor to execute an Applesoft subroutine located at $D823. This particular subroutine happens to be the so-called "running return" to Applesoft, after which the computer will begin to execute whichever proper Applesoft command it encounters next.

You'll notice, however, that so far this Monitor command line is still resident in a string variable; how do we get the Monitor to see it and execute it? Well, line 70 pokes this string, one byte at a time, into memory starting at location 512 (in decimal). But 512 is equivalent to $0200, the start of the Apple's keyboard input buffer where it goes to find every new line after you have typed it in. So the effect of line 70 is to place the pseudo-input line defined in line 60 into the input buffer. Those who are particularly observant may be wondering about the reason for adding 128 to the value produced by the ASC command, before POKEing it into the buffer. This is due to a little known incompatibility between Applesoft and the Monitor in the way they interpret ASCII character codes. Strangely enough, although Applesoft uses "true ASCII," in which the highest bit (bit 7) of each byte is off (i.e., =0), the Monitor uses a different version of ASCII in which bit 7 of each byte has to be on (i.e., =1). The addition of 128 (decimal) turns this bit from off to on.

Now line 80 gets the Monitor to look into the keyboard buffer and execute whatever commands it finds there. The POKE of 0 into location 72 is just a precaution, to make sure that no strange values have gotten into the location which will be stored in the Processor Status Register when the Monitor call is executed. Those of you who know something about the operation of the 6502 Microprocessor will understand what this means; for the rest of you it is of no great significance — it just needs to be done to prevent possible trouble. Finally, the command CALL-144 jumps to the Monitor subroutine referred to above: the one that scans the input buffer and executes whatever commands it sees there.

As I mentioned above, lines 60-80 are the heart of the technique being discussed in this article. But I want to emphasize that the procedure outlined in the past few paragraphs is *extremely* powerful and quite general. By using it, you can make the Apple execute any commands which can be input by way of the Monitor, such as moving ranges of memory around, storing machine language pro-



16K RAM EXPANSION BOARD FOR THE APPLE II* $195.00

The Andromeda 16K RAM Expansion Board allows your Apple to use RAM memory in place of the BASIC Language ROMs giving you up to 64K of programmable memory. Separate Applesoft* or Integer BASIC ROM cards are no longer needed. The 16K RAM Expansion Board works with the Microsoft Z-80 card, Visicalc, DOS 3-3, Pascal, Fortran, Pilot, and other software. A switch on the card selects either the RAM language or the mainboard ROMs when you reset your Apple.

The Andromeda 16K RAM Expansion Board has a proven record for reliability with thousands of satisfied customers.

Now with One Year Warranty.

*Apple II and Applesoft are trademarks.

ANDROMEDA
INCORPORATED
P.O. Box 19144
Greensboro, NC. 27410
919 852-1482

Distributed By:

COMPUTER DATA SERVICES
P.O. Box 696
Amherst, NH. 03031
603 673-7375

grams wherever you wish, or any of the other things discussed on pages 39–60 of the *Apple II Reference Manual*. But you can do all this from *within* a running Applesoft program, without ever stopping to enter the Monitor or typing in any commands at the keyboard. To those with a fertile imagination, the possibilities inherent in such a capability are enormous — enough to fill several articles as long as this one. You can have some fun thinking up some ideas of your own.

Meanwhile, let's get back to the subject at hand. After executing line 80, the computer should have loaded the ML program from tape into the specified location in memory. Line 90 checks the horizontal position of the cursor after the load has been completed. If the loading failed, the computer will have printed out the message "ERR", and so the cursor will be three spaces farther to the right of where it was before the loading process began. In this case, line 90 causes a jump to line 200, the error exit. Here the "bell" is beeped thrice (those CHR$(7)s) to wake me up, an appropriate message is printed on the screen, and the program ends, to let me rewind the cassette and try again.

But this doesn't happen very often — the Apple cassette system has been very reliable for me. So, usually, upon completing the tape load, the program goes to line 100. This is another very important line whose significance, however, cannot be easily explained at this point in the discussion. Let us put off the explanation of line 100 until we have finished looking at the remainder of the program. There's not much left to say. Line 110 lets me know, with a "beep" and a message, that the loading process has been successfully completed and reminds me to turn off the tape recorder. Line 120 causes a delay of about three or four seconds to give me time to see and act on that message, because line 130 causes the program to begin executing.

This may need a bit of comment. Although it is necessary, in Applesoft, to RUN to start a program after loading it, I think that most of the time the user would be just as happy to have the program begin running as soon as the load was completed, if only Applesoft had such a "LOAD-AND-GO" command. Certainly in the present example, since I know that the entry address to initialize my Assembler is at $1000 (decimal 4096), I prefer to have the loader program do this for me by doing a "CALL 4096". You can "Load-and-Go" your own ML programs in the same way by putting an equivalent CALL to the entry address in your version of this loader.

However, if you insist on retaining the two-step process, and want to be able to start your program by typing RUN in the regular Applesoft manner, the program can easily be modified to do this instead. Just replace line 130 with the following:

```
130 DEL 10,130: END
140 CALL 4096: END
```

The new line 130 causes the whole front part of the loader to self-destruct (in memory only of course, not on your cassette), leaving only line 140 as the first active command. Now typing RUN executes just this one remaining line, making your ML program start running at its entry address.

This has been a rather exhaustive description of a short Applesoft program, but since it contains several techniques which may be new and unfamiliar to many readers and since these techniques seem to me to be of great usefulness, I thought it worthwhile to explain thoroughly.

One Of Applesoft's Least-Known Features

But we're not quite done explaining yet. There is one more technique which is required to make the loader perform properly. And this is perhaps the most mysterious and least-known of all the features of Applesoft, so even some of you semi-pros might be able to learn something new from the next few paragraphs.

As things now stand, the loader program and your ML program have not yet been joined together on tape so that the former can help you to load in the latter. You will recall that, before I started describing the loader program, I left you with your ML program already in memory. Now you should also have typed in a customized version of the loader program, with the beginning and ending addresses in string Y$ replaced by the values appropriate to your ML program, and your entry address (in decimal) replacing my "4096" in line 130 (or 140 if you chose to go that route). (By the way, I hope that your ML program didn't occupy any of the memory spaces now containing the loader (from $0800 to $09A2 in my case), since I forgot to warn you about this unfortunate way to clobber the whole thing.) Assuming that all is still well, you now want to put both the loader and the ML program onto tape, with the loader first, of course. But, before you hasten to type SAVE to put the Applesoft loader program on tape, wait just a little longer while I explain the last secret.

The secret is this: *before* you SAVE the loader, type in the following Applesoft command in Immediate Mode (to be executed from the keyboard):

```
POKE 82,128
```

This seemingly innocuous command is the key to making the loader behave like an EXEC file — doing its job without human intervention. It represents an almost totally undocumented feature of Applesoft and works like this: any Applesoft program which is SAVEd to tape after this POKE has first been executed will AUTO-RUN as soon as it has been LOADed! That is, if you rewind the SAVEd tape and type LOAD, Applesoft will not only load in the program, but will also immediately begin running it *without waiting for you to type RUN.* So Applesoft does, after all, have an AUTO-RUN command; you just have to know how to get at it.

Now the background for making the Tape-Exec loader is complete. Do the POKE, then SAVE the loader program, then enter the Monitor and SAVE your ML program by typing:

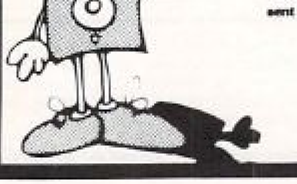
(Begin Address).(End Address)W

in the usual way. (In the case of my Assembler, for example, I used 1000.24FFW to SAVE it.) Now try it out. Rewind the tape, and type LOAD in the good old Applesoft way. The loader will be loaded and will immediately begin to run by itself, causing your ML program to be loaded too, in accordance with the instructions placed in its Y$ string by you. From now on, it will be as easy to load this ML program as any Applesoft program.

There is just one potential problem with all this, but I have taken care of it by the as-yet-unexplained POKE in line 100 of the loader. You see, the magic words "POKE 82,128" which you invoked before SAVEing the loader constitute a much more powerful spell than I have yet indicated. They do more than just cause an Applesoft program (in this case, the loader) to Auto-Run. They also completely lock up Applesoft so you can't use it very much. It will allow you to RUN the program in memory, but any other valid or invalid Applesoft command will be ignored. You won't be able to LIST, SAVE, alter, or do anything else to the program as long as the effects of that POKE remain active. This is a very powerful magic you have invoked here, but it would take us too far afield from the main topic to explain all its ramifications now.

Fortunately, however, it is not hard to undo the effects of that magic from within the loader program itself (although quite difficult, and sometimes impossible, to undo it from outside a running program!) Line 100 is the required antidote. It leaves everything just as you are accustomed to having it in an Applesoft environment. ©

(Continued in next issue.)



Dysan
CORPORATION

Solve your disc problems, buy 100% surface tested Dysan diskettes. All orders shipped from stock, within 24 hours. Call toll FREE (800) 235-4137 for prices and information. Visa and Master Card accepted. All orders sent postage paid.

PACIFIC
EXCHANGES
100 Foothill Blvd
San Luis Obispo, CA
93401 (In Cal call
(805) 543-1037)

SEPTEMBER IS "TAKE CARE OF BUSINESS" MONTH

VISICALC

Take any problem you would normally work out on paper in rows and columns, use VISICALC and see why it is probably the single most valuable program yet developed for personal and professional use. VISICALC, the electronic worksheet that performs arithmetic calculations INSTANTLY.

Regular Price: $199.95
September Special: $174.95
Cat No. 2718 Apple II/II + , 32K disk
Cat No. 2808 Atari 800, 32K, disk

The Home Money Minder

Been looking for an easy to use home management tool? This program includes: checkbook balance, budget analysis, income and expenditures with graph comparisons and more.
Regular Price: $34.95 September Special: $29.95

Cat No. 2938 Apple II + , Applsft, 48K disk

How To Order

Prices include UPS ground shipping & handling (USA only). Pay by check, M.O., Visa, M/C or C.O.D. ($1.40 ADDL) Mention this AD to qualify for special prices. Offer expires 10/1/81.

HW ELECTRONICS

19511 Business Center Dr., Dept. G9
Northridge, CA 91324

(800) 423-5387 (Outside Calif.)

(213) 886-9200 (In Calif.)

Send for our FREE catalogue Today!

Text Composition On The Apple II Plus

R. R. Hiatt, John Rustenburg and
Stefan Demmig
St. Catharines, ONT

Text composition on the Apple II Plus presents two problems, interfacing to some sort of printer and distinguishing between upper and lower case on both CRT and printer output. The first of these is readily solved by the Apple II Reference manual: We are pleased to report that the circuit given as Figure 1 on p. 118 requires no modification for interfacing a Decwriter II to the Apple II Plus. The software (TTYDRIVER, p. 119), does require a small change: deleting the text window width setting to 72 (replacing the code in $378-$37B with NOP's). This avoids the system crash that results when control is returned to the CRT with a text window greater than 40. Furthermore, text window setting is more flexible when incorporated in the BASIC calling routine.

Upper vs. lower case with the standard Apple keyboard is trickier. The shift key is live only for dual function keys such as @/p and ?/N. The CTRL key is also dual purpose; e.g., if CTRL M were to be interpreted as cap M, there would be no unambiguous signal for carriage return. Fortunately, the ESC key can be made to suit the purpose after a bit of fooling around to see how it affects code received at the keyboard inport ($C000).

Programs 1 and 2 are short ESC demo routines. ESCDEMO1 shows the transient nature of the ESC effect. (Key ESC; then before all 16 27's are printed, key a letter.) ESCDEMO2 is a little more amusing. The ESC, (CHR$(27)), is captured in an apparently infinite loop. Subsequent keying of a letter, however, breaks the loop and results in a print of *both* "UPPERCASE" and "LOWERCASE," as if both of the two mutually exclusive IF clauses were being followed. (Of course, they are, but not as it immediately appears. We leave it to the reader to figure out the logical paradox.)

Program 3 gives a simple text composition routine, employing both DECWRITER (our name for the modified TTY driver) and ESC for upper case letters. The main program, starting at line 400, augments the routine of Program 2 by capturing the ESC'ed ASCII code and then resetting the keyboard strobe. (Resetting the strobe first de-ESC's the value.) To facilitate corrections, the text is echoed to the CRT, with left arrow (←) activated for erasure, and is sent to the printer only after a <CR> (end of line).

Training oneself to use ESC for upper case, rather than shift turns out not to be as difficult as it might seem, as long as upper case letters are distinguished on the CRT in some way. We have taken the route of setting upper case to FLASH via the code in the subroutine at 200.

The single character FLASH requires a POKE at the appropriate screen memory address. While the base vertical address can be worked out from the vertical cursor position (PEEK(37)) and a base 8 algorythm, it turns out that the base address for TEXT/LORES graphics is easily obtained by PEEK(40) + 256*PEEK(41). Adding PEEK(36) (horizontal) to this gives the cursor position.

Obviously Program 3 is not a text editor or even a proper front end for one. It does, however, solve what we have felt to be the major problems — those involving the system. The rest is simply a matter of creative BASIC.

Program 1. ESCDEMO1

```
10 GET Q$
20 FOR I = 49152 TO 49167
30 PRINT PEEK (I), I: NEXT: GOTO 10
```

Program 2. ESCDEMO2

```
10 P = 49152
20 GET Q$
30 IF PEEK (P) = 27 THEN 30
35 CH = PEEK (P)
40 IF CH > 127 THEN PRINT "UPPER CASE":
   GOTO 20
50 IF CH < 128 THEN PRINT "LOWER CASE":
   GOTO 20
```

```
10 REM SECRETARY
20 DIM LN% (200)
30 D$ = CHR$ (4): B$ = CHR$ (7): P =
   -16384: Q = -16368
40 PRINT "PROGRAM FOR WRITING HARD COPY
   TO DEC II": PRINT: INPUT "WANT
   INSTRUCTIONS ? "; Q$: IF Q$ > = "Y"
   THEN GOSUB 600
50 INPUT "LINE LENGTH FOR DEC II ? "; LW
60 IF PEEK (880) < > 169 THEN PRINT D$;
   "BLOAD DECWRITER"
70 GOTO 400
100 REM SBR BACKSPACE
110 IF LL = 0 THEN RETURN
120 LL = LL -1
130 PRINT CHR$(8);
140 RETURN
200 REM SBR FLASH CAPS
210 HZ = PEEK (36)
220 PN = 256 * PEEK (41) + PEEK (40) +HZ
230 POKE PN, CH - 128
240 POKE 36, HZ + 1
250 RETURN
300 REM SBR WRITE TO DEC II
310 CALL 880
320 POKE 33, LW
330 FOR I = 1 TO LL
340 PRINT CHR$( LN%(I));
350 NEXT
360 PRINT
370 POKE 33,40: PR# 0
380 PRINT
390 RETURN
400 REM CHARACTER INPUT
410 CALL - 936
420 LL = 0
430 UC = 0
440 GET Q$: CH = PEEK (P)
470 IF CH = 8 THEN GOSUB 100: GOTO 430
480 IF PEEK (P) = 27 THEN UC = 1: GOTO
    480
490 CH = PEEK (P)
500 POKE Q, 0
510 IF CH > 64 AND CH < 91 THEN CH = CH +
    32
520 LL = LL + 1: IF LL = LW - 8 THEN
    PRINT B$;
530 IF LL > LW THEN LL = LL - 1: GOSUB
    300: LL = 1
540 LN%(LL) = CH
550 IF UC THEN GOSUB 200: GOTO 430
560 PRINT Q$;: GOTO 440
600 REM INSTRUCTIONS
610 HOME : PRINT "TYPE NORMALLY, BUT USE
    ESC KEY FOR
620 PRINT "UPPER CASE LETTERS>  (UPPER
    CASE ON": PRINT "SCREEN IS SET TO
    FLASH)": PRINT
630 PRINT "THE SHIFT KEY IS STILL USED
    FOR UPPER": PRINT "SYMBOLS ON DUAL
    FUNCTION KEYS": PRINT
640 PRINT "TO END PROGRAM KEY CTRL A"
650 FOR I = 1 TO 3000: NEXT: PRINT:
    RETURN
```
©

TOLL FREE
Subscription
Order Line
800-227-1617
In CA 800-772-3545
Please ask for Extension 401.

★★★★★★★★★★★★★★★★★★★

Bugs in your Apple?

DDT

Disco-Tech's Disc Drive Timer program
zaps disc drive problems!

Analyze disc drive motor speed on a routine basis with an adjustable real-time speedometer. Accurate to one-tenth of one r.p.m. out of 300 r.p.m.

Fine-tune disc drive motor speed yourself. All you need is DDT, two screwdrivers, and five minutes' time.

$29.95 Postpaid
California residents add 6% tax
Diskette & complete manual

Engineering Business
Architecture Utilities
Surveying

Also available for TRS-80 Model I.

To order or for more information,
write or call

disco tech T.M.
microcomputer products
a division of
Morton Technologies, Inc.
600 B Street
P.O. Box 11129 • Santa Rosa, CA 95406
707/523-1600
Dealer inquiries invited

★★★★★★★★★★★★★★★★★★★

Algebra String
A Self-Altering Program For The Apple-II

Winston Cope
St. Petersburg, FL

BASIC is essentially an arithmetic language. Its symbol manipulating capability is used mainly to provide conveniences for the user, to provide instructions for the user, or to give headings. An algebraic expression is part of the program text, and is considered a calculation.

There is no easy way to operate on a mathematical expression itself, for example, to take a derivative. A program must be written which inputs a mathematical expression as a string and yields another string as output. Applesoft provides string manipulation commands which make this possible. The expression is still a string, however, and there is no easy way to derive numbers from it, to graph it, for example.

"ALGEBRA STRING" is a demonstration of how a mathematical string expression may be transformed into an arithmetic variable expression which can be used by the program. The concept behind this program is to take an algebraic string expression, Y$, to expand it to a standard length, and to poke it back into the program text itself at the proper position. Care must be taken to translate operation symbols, such as +, into their token form.

This program considers a function Y of X, and subroutine 2000 performs a simple listing of an array Y(X), for X going from 1 to N. Subroutine 62100 inputs the expression as Y$, and expands to a length of 50 characters, by concatenating "+"s and a final "0". Subroutine 62200 takes this expanded expression and POKES it into memory so that it appears at the proper place in the program text, here, at step 1020, beginning at memory location Z. Arithmetic operators are represented in strings as ASCII, but have token values when used for calculation, so this subroutine performs these substitutions. The arithmetic expression in the program whose place is taken by Y$, in step 1020, must have the same length as Y$, here 50.

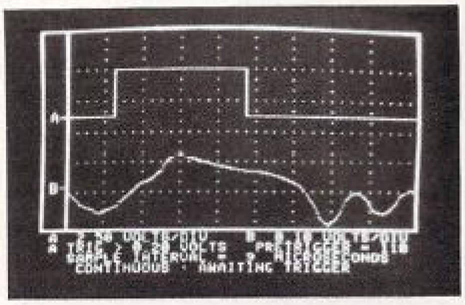
Subroutine 62000 determines Z. This is simply done by finding the memory location which contains a "+." such that the next 5 locations also contain "+." The odds are very small that this would happen anywhere else than Z. LO and HI could be 0 and 64000, but for a particular program one can markedly narrow the range of the search.

When the operator is finished entering expressions to evaluate, the program will initial-

APPLESCOPE

DIGITAL STORAGE OSCILLOSCOPE

Interface for the Apple II Computer



The APPLESCOPE system combines two high speed analog to digital converters and a digital control board with the high resolution graphics capabilities of the Apple II computer to create a digital storage oscilloscope. Signal trace parameters are entered through the keyboard to operational software provided in PROM on the DI control board.

- DC to 3.5 Mhz sample rate with 1024 byte buffer memory
- Pretrigger Viewing up to 1020 Samples
- Programmable Scale Select
- Continuous and Single Sweep Modes
- Single or Dual Channel Trace
- Greater than or less than trigger threshold detection

Price for the two board Applescope system is **$595**

For futher information contact: **RC ELECTRONICS INC.**
7265 Tuolumne Street
Goleta, CA 93117
(805) 968-6614

Dealer Inquiries Invited

CLASS OF '82 . . . BOOT UP!

ASSISTANT PRINCIPAL - Whether you're too small for a big computer or if you just want to bring your administrative duties under your control, the "Assistant Principal" is just what you need. It prints school rosters, allows you to input teacher grades, prepare report cards, and maintain student master records. You should have no more trouble with student scheduling, grade averaging or recording of student grades. For Apple II and Apple II Plus, two disk drives, 48K Memory and an 80 column printer. Item M-3839-11, Price $500.00.

ROLL CALL - Let this program keep track of that all important attendance data. Whether you need information on an individual's tardy, absence history or need to prepare attendance reports for the board or your state funding agency, this complete system will make the task easy work. The system will create its own data base or work off the Assistant Principal's files. For Apple II or Apple II Plus, two disk drives, 48K Memory, and an 80 column printer. Item M-52. Price $250.00.

SYSTEMS ANALYST PLUS - Whether you're a novice computer student, teacher or principal, you will find this program most helpful. Even the advanced programmer should find these seventeen teaching sessions on programming useful for brushing up on the language and commands. For Apple II or Apple II Plus with Applesoft, one disk drive, and 32K Memory. Item M-40. Price $69.95.

Call our toll free number for a free catalog on professional, business, accounting and educational programs.

(800) 854-0561
In Ca. (800) 432-7257
Ext. 802 or write:

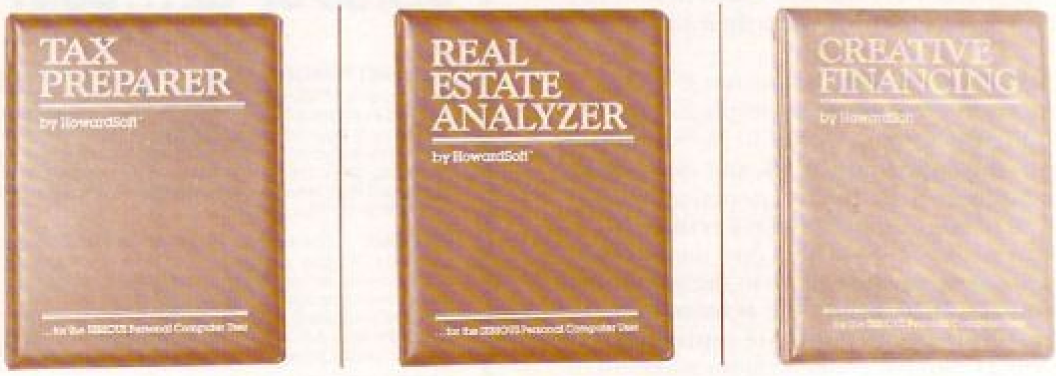
MONUMENT COMPUTER SERVICE
Village Data Center
P.O. Box 603
Joshua Tree, California 92252

ize itself by filling up the expression at 1020 with "+" 's so it can be run again.

```
5 ALGEBRA STRING
10 PRINT "ENTER # OF POINTS": INPUT X
100 LO=2000:HI=2500
105 DIM Y(N)
110 GOSUB 62000
115 GOSUB 1000:GOSUB 2000
120 PRINT "ALL DONE? (Y/N)":GET C$
125 IF C$="N" THEN 115
130 Y$="++++++": GOSUB 62200
135 END
1000 REM LOAD ARRAY Y(X)
1005 PRINT "ENTER ALGEBRAIC WITH
DEPENDENT VARIABLE X":INPUT Y$
1010 GOSUB 62100:GOSUB 62200
1015 FOR X=1 TO N
1020 Y(X) = + + + + + + + + + + + + + + +
+ + + + + + + + + + + + + + + + + + + + +
+ + + + + + + + + + + + + + + + + + + + +
+ + + + + + + + + + + + + + + + + + + + +
+ 0
1025 NEXT X
1030 RETURN
2000 REM DISPLAY Y(X)
2005 J=0
2010 FOR X=1 TO N
2015 PRINT X,Y(X):J=J+1
2020 IF J=20 THEN PRINT:PRINT "PRESS ANY
KEY":GET C$:J=0
2025 NEXT X
2030 RETURN
62000 REM FIND Z
62010 FOR J=0 TO 5
62015 IF PEEK(I+J) <> 200 THEN 62030
62020 NEXT J
62025 Z=I: RETURN
62030 NEXT I
62035 PRINT "CANNOT FIND Z": END
62100 REM STANDARD LENGTH FOR Y$
62105 IF LEN(Y$)<49 THEN Y$=Y$ + "+":GOTO
62100
62115 RETURN
62200 REM SUBSTITUTE Y$ INTO PROGRAM TEXT
62205 FOR I=1 TO LEN(Y$)
62210 Q= ASC(MID$(Y$,I,1))
62215 IF Q=43 THEN Q=200
62220 IF Q=45 THEN Q=201
62230 IF Q=47 THEN Q=203
62240 IF Q=94 THEN Q=204
62245 NEXT I
62250 RETURN
```

©

A NEW standard in software QUALITY . . .



. . . for the SERIOUS personal computer user

These packages will turn your Apple Computer into a powerful financial tool previously accessible only to the largest corporations. These packages are so unique in the completeness and sophistication of both the software and the documentation that they have been acclaimed by business professionals everywhere. Yet the software is so friendly that it can be operated by the financial or computer novice. See for yourself at your local dealer.

HSS Howard Software Services
...for the SERIOUS Personal Computer User
6713 Vista del Mar | La Jolla, CA 92037 | (714) 454-5079

WHY THE MICROSOFT RAMCARD™ MAKES OUR SOFTCARD™ AN EVEN BETTER IDEA.

Memory — you never seem to have quite enough of it.

But if you're one of the thousands of Apple owners using the SoftCard, there's an economical new way to expand your memory dramatically.

16K ON A PLUG-IN CARD.

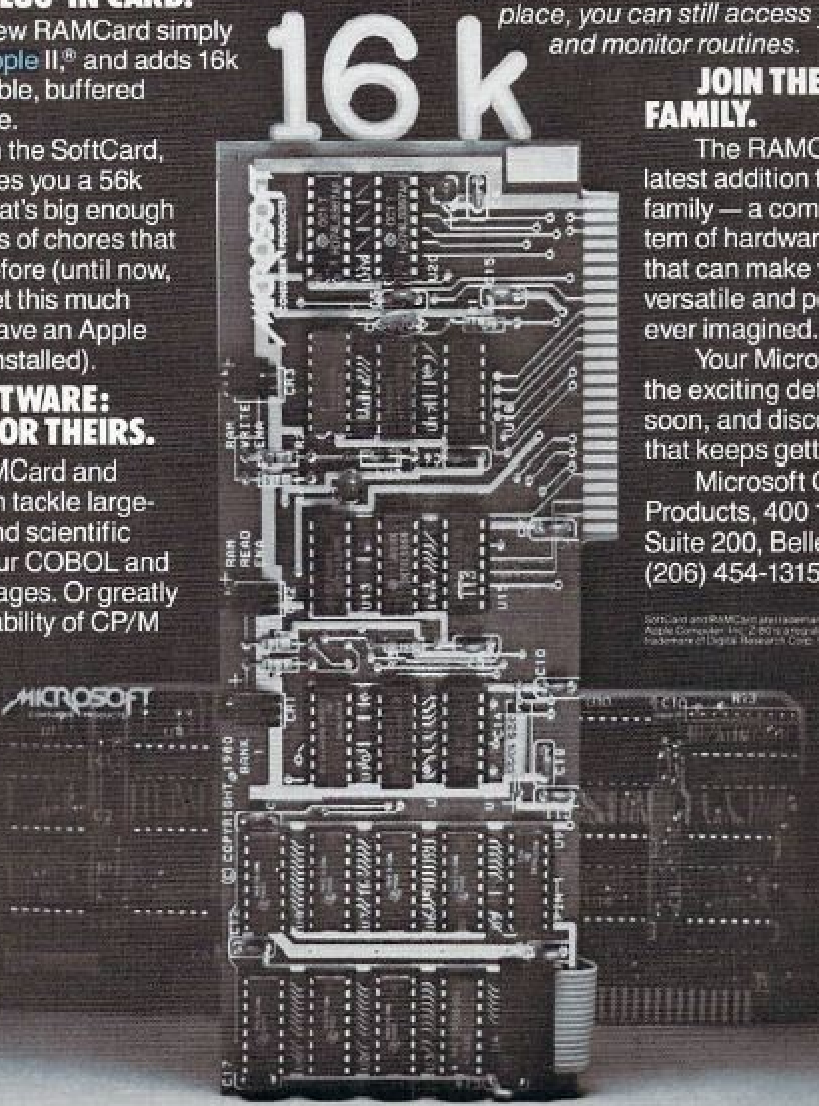
Microsoft's new RAMCard simply plugs into your Apple II,® and adds 16k bytes of dependable, buffered read/write storage.

Together with the SoftCard, the RAMCard gives you a 56k CP/M® system that's big enough to take on all kinds of chores that would never fit before (until now, the only way to get this much memory was to have an Apple Language Card installed).

GREAT SOFTWARE: YOURS, OURS, OR THEIRS.

With the RAMCard and SoftCard, you can tackle large-scale business and scientific computing with our COBOL and FORTRAN languages. Or greatly increase the capability of CP/M applications like the Peachtree Software accounting systems. VisiCalc™ and other Apple software packages can take advantage of RAMCard too.

And RAMCard gives you the extra capacity to develop advanced programs of your own, using the SoftCard and CP/M. *Even with the RAMCard in place, you can still access your ROM BASIC and monitor routines.*



JOIN THE SOFTCARD FAMILY.

The RAMCard is just the latest addition to the SoftCard family — a comprehensive system of hardware and software that can make your Apple more versatile and powerful than you ever imagined.

Your Microsoft dealer has all the exciting details. Visit him soon, and discover a great idea that keeps getting better.

Microsoft Consumer Products, 400 108th Ave. N.E., Suite 200, Bellevue, WA 98004. (206) 454-1315.

SoftCard and RAMCard are trademarks of Microsoft. Apple II is a registered trademark of Apple Computer, Inc. Z-80 is a registered trademark of Zilog, Inc. CP/M is a registered trademark of Digital Research Corp. VisiCalc is a trademark of Personal Software, Inc.

MICROSOFT

Undeletable Lines

Michael P. Antonovich
Wyomissing, PA

Editor's Note: Though described for Applesoft, this crafty technique works on CBM computers as well. For PETs, type SYS1024. Then .M 0400 041F and hit RETURN (where the author mentions CALL-151). — RTM

Did you ever wish that you could put your name into a program in such a way that the average computer user could not delete it and claim the program as his own? Well you can. In APPLESOFT, you normally cannot enter program lines greater than 63999, but I will show you how you can. First we have to see how the APPLE stores your program lines in memory.

The APPLE stores APPLESOFT program lines beginning at memory location $800. (The '$' sign before a number indicates that the number is in hexadecimal). Let's enter the following small program to illustrate the way a program is stored.

```
1 REM
2 A=8
3 PRINT A
4 END
```

To see how the APPLE stores this program, we have to enter the monitor with a CALL -151. However, before we list the program, there is one other piece of information that we need to determine. To add lines to an existing program, you need to know where the current program ends in memory. You can accomplish this in one of two ways. You can page through the memory to find the program's last byte. (Hint: This is the hard way.) The APPLE also stores the location of the last memory byte in locations $69 and $6A. Let's enter the monitor now to check our program.

```
CALL -151
*69.6A
0069-1E 08
*800.81F
0800-00 07 08 01 00 B2 00 0F
0808-08 02 00 41 D0 38 00 16
0810-08 03 00 BA 41 00 1C 08
0818-04 00 80 00 00 00 FF FF
```

Although you may not recognize it, the above listing is a memory dump of your program. Let's examine how our BASIC lines were translated to the above hex dump.

The first byte $00 at location $800 has no special meaning to our program. In fact, location $800 will always contain $00. The program lines begin after this point. Each line is prefixed by four bytes. The first pair of bytes stores the starting byte address of the next line. In our example, locations $801 and $802 indicate that the next line will begin at memory location $807. (Remember that the location is split into two bytes which are stored in what seems, to us humans, to be in reversed order.)

The second pair of bytes contains the line number assigned to the program line. In our example we started with the line number "1." Thus memory locations $803 and $804 indicate that the first line number is "1." In addition to the four bytes which prefix each line, each line is ended with single byte '00' to separate it from the next line. Therefore, there is a five byte overhead for each program line used. If multiple statements are combined with a colon (using one byte) on a single line, you can save four bytes for each extra line you eliminate. If you have any doubts, try it yourself with the above program.

... though described for Applesoft, this crafty technique works on CBM computers as well.

The second program line begins at memory address $807. The first four bytes indicate that the next statement will begin at location $80F and will have statement number "2." The next three bytes "41 D0 38" represent the tokens for the equality: A=8. The information we will need to understand these tokens is found in Appendix F and Appendix K of the APPLESOFT Reference Manual. I'll wait while you go get your copy.

Appendix F lists the decimal tokens for all of the keywords used by the APPLE. However, when we are in the monitor, we need the hexadecimal equivalent of the tokens. For example, the hex equivalent for END is $80, REM is $B2, and PRINT is $BA, etc. You might want to take the time now to write the hexadecimal equivalents next to the decimal values for all of the tokens.

Variable names, numbers and strings are not listed in Appendix F. These must be constructed by using the individual ASCII character representations. Appendix K lists the ASCII character set with the decimal and hexadecimal codes. Again, we are interested in the hexadecimal codes. In our example, we need the "A" or $41 and the "8" or $38.

I have now explained all of the hex codes used in our example except one, the equal sign (=). The problem is that both Appendix F and K contain an equal sign, but each has a different hex code. Which one is correct? We need a rule which will guide us with this and similar problems, so here it is. To construct a variable name, number, or string of characters, use Appendix K. Any symbol used in an arithmetic expression (such as =, (,), etc.) should be taken from Appendix F.

Finally, even though we end our program with an END statement, the APPLE does not know that this is the end of the program (for, indeed, it does not have to be). The end of the program is indicated by the byte pair '00 00' in the locations where it expects to find the next line number.

Now that we know how the APPLE interprets our APPLESOFT program and stores it in memory, we are ready to begin adding lines to our program which will be undeletable to the average APPLE user.

Normally, only program line numbers from 0 to 63999 can be used and referenced in an APPLESOFT program. Converting 63999 to hexadecimal we get $F9FF, but we *can* write larger hexadecimal numbers than in two bytes. In fact, we should be able to use the numbers from $FA00 through $FFFF or 64000 through 65536. Even though the APPLE will prevent us from entering these line numbers through the keyboard, we know enough about how the APPLE stores program lines to sneak them in.

Let's keep this example simple and assume that I just want to store my name and the date as remark statements. We could just as easily make them PRINT statements if we wanted to. The statements we want are:

```
64000  REM MICHAEL P. ANTONOVICH
64001  REM JUNE 28, 1981
```

Now enter the monitor (CALL -151) and type the following:

```
81C:37 08 00 FA
820:B2 4D 49 43 48 41 45 4C
828:20 50 2E 20 41 4E  54 4F
830:4E 4F 56 49  43 48 00 4A
838:08 01 FA B2 4A 55 4E 45
840:20 33 30 2C 20 31 39 38
848:31 00 00 00
```

Before you return to APPLESOFT, we must reset the end of program pointer. If we do not do this, the first time you run your program, any variables you store will write over the new lines you just added. Our program now ends at memory location $84C. This information must be put into locations $69 and $6A.

```
69:4C 08
```

We can now reenter APPLESOFT using CNTL-C RETURN, and list the program. There are lines 64000 and 64001 at the end. Try to delete them. HA! You cannot touch them. You can SAVE this program, reload it, RUN it, and copy it, and still those two lines will be there. In fact the only way to get rid of them is to enter the monitor, find where you want the program to end, change the last two bytes to "00 00," and change the program ending location in addresses $69 and $6A. It's easy, but only if you know how.

Remarks are not the only things that you can put into undeletable lines. You can store anything you want from program lines using tokens and character strings to machine language programs using this method.

I would like to clear one thing up before anyone gets the wrong idea. The tokens we have been using above are NOT machine language. All microcomputers use tokens to store keywords. All BASIC program lines are stored in the above manner, not in machine language. The program lines must be interpreted each and every time that they are run. However, there is a lot that you can do with undeletable lines.

[Manipulating BASIC using your monitor can also hide lines from LISTing, but they will RUN normally. To make this program print the 8, but without *line 3 appearing in the program LISTing — simply change the one hexadecimal number to 16 like this:*

```
.: 0800 00 07 08 01 00 B2 00 16
        Apple Version
.: 0400 00 07 04 01 00 8F 00 16
        PET Version
```

Also notice that PET and Apple versions are similar (both are Microsoft BASICs), but the hex number for REM is $B2 for Apple, $8F for PET. The location of the last BASIC memory byte (the top of a BASIC program) is different in the PET, too. For Original PETs, this address is found in $007C, $007D. For Upgrade and 4.0 PETs: $002A, 002B. Another difference is that Apple starts its BASIC programs at $0800, where PET starts at $0400. You will notice this reflected in the line links which contain the starting byte address of the next line. (In PET, the first links are at $0401 and $0402, and they point to $0407 where the next link points up the chain once again. It is this $0407 (or Apple's $0807) which we changed to skip over line three in our "hidden line" technique above.) Here is the complete program as it would appear in the PET:]

```
.: 0400  00  07  04  01  00  8F  00  0F
.: 0408  04  02  00  41  B2  38  00  16
.: 0410  04  03  00  99  41  00  1C  04
.: 0418  04  00  80  00  37  04  00  FA
.: 0420  B2  4D  49  43  48  41  45  4C
.: 0428  20  50  2E  20  41  4E  54  4F
.: 0430  4E  4F  56  49  43  48  00  4A
.:  0438  04  01  FA  B2  4A  55  4E  45
.:  0440  20  33  30  2C  20  31  39  38
.:  0448  31  00  00  00  AA  AA  AA  AA
```

©

THE apple® GAZETTE

Budgeting On The Apple

William R. Swinyard
Provo, UT

When I sat down at one of the University's dozens of Apples six weeks ago, someone had to show me where the on/off switch was. But I learned fast and, within a few weeks, I was doing pretty well at what I believed the Apple to be best at. I had maintained several "high score" records on Apple Invaders ... I was terrific at judging angle and windspeed in the Bombs game ... and I was even learning how to back away from brick walls in Maze. But I had not entered even one real line of programming.

One other thing I did discover, though, was how ... well, *nice* the Apple is. True, I had spent a little programming time on our DEC installation (and a *lot* of time on it using canned analytical software packages), but I always felt inhibited about getting too intimate with that machine because of its so formal, even intimidating, way of speaking — or should I say "shouting" — back. "FATAL PROCESSING ERROR. ILLEGAL CHARACTER IN DATA (K) [OCTAL 113] UNIT=8/ACCESS= SEQUIN/MODE=ASCII," was typical of the helpful way the DEC had about it. What a contrast with the Apple! It almost seemed to *invite* me to stay with it just a bit longer, and even to try running a few little programs on it.

And so I did. This article is about one of the first. It simply permits the user to enter a monthly income figure then displays a budget, along with the discrepancy between the budget and the income. It has a few nice touches, I think. Program requests for budget changes are highlighted in a new budget display, there is a short subroutine to align the decimal points, and percentages of total budget are also displayed.

Briefly, here are what the sections of the program do:

Line 115 — Displys the title of the program on an otherwise blank screen for a few seconds.

Line 120 — Requests monthly income information.

Lines 240-570 — Lists out, with illustrative budget figures, the items appropriate for *my* budget. Yours may be different, and you will need to change the program listing where appropriate.

Lines 600-680 — Prints the output headings, the budget name and corresponding value, the total budget figure, and the deviation between income and budget.

Lines 690-775 — The above information is displayed for about 10 seconds, followed by requests for updated expenditures originating in these lines. If updates are made, a new display table is printed which highlights (with underlining) the updated expenditure figure.

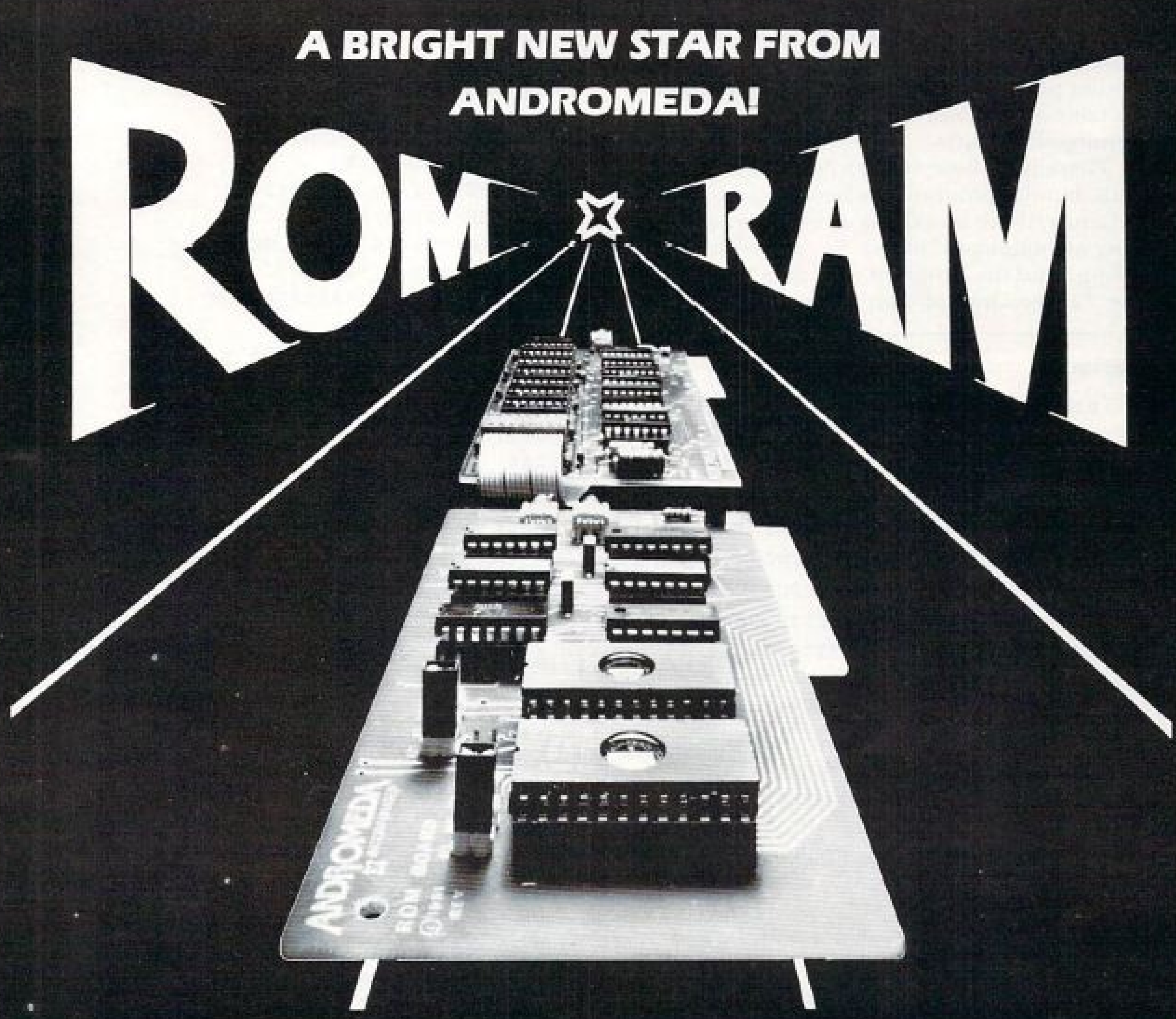
Lines 780-830 — A subroutine for lining up the decimals.

Lines 835-900 — Prints out the fully revised expenditures table, but accompanied this time with percentage allocations of budget, if requested.

The tabular display has been arranged to just fit into the Apple screen, so a line printer is unnecessary to get full use of this program.

Several refinements to the program might make it yet more appealing for your use, but seemed like overkill for me. With little modification the program could be designed to display budget values and percentages in the first tabular display — only one extra FOR-NEXT loop would be required. The program could easily be rewritten to make it interactive in budget-area development, though this seems unnecessary for a family whose budget allocations typically fall into the same areas each month. Or the program could be easily modified to accommodate input of *gross* monthly income, and incorporate federal and state tax withholding information, FICA withholding, tax-deferred income, etc. by building in the appropriate withholding proportions (income level, number of exemptions, and claimed marital status would be required as input values). However, since Uncle Sam has been changing the FICA rates annually, I passed this improvement by in favor of the relative permanency of the program as it now stands. Still, if you are interested in simulating the effect of increasing or decreasing your number of exemptions, this modification could be worthwhile.

Figure 1 shows the initial output display after entering a net monthly income figure. Figure 2 shows the output following the program's request for any changes. You can see that the change was



NEW ROM BOARD FOR THE APPLE II*
$125.00 WITH UTILITY ROM.

With Andromeda's new ROM Board, you can plug many useful utility programs into your Apple II. Because ROM memory never forgets, you can access these utilities instantly without having to load them from disk.

The ROM Board comes with the utility ROM, which gives you five powerful options to apply to your Applesoft* programs. With the Utility ROM, you can do automatic line numbering, control a program list with a page mode, restore a crashed Applesoft* program in memory, alphabetize a disk catalogue and create a disk without DOS, giving you an extra 8K on your disk. Any of Soft Control Systems' other ROMS can be used, such as the Dual DOS in ROM, and Your'ple ROM.

You can install 2K PROMS, 4K PROMS, or even 2K RAM chips in each of the two memory sockets. So you can even have the Read - Write capability of RAM to develop PROM Programs yourself, or just have an extra 2K RAM for your machine - Language programs. Two 2732 PROMS allow a total of 8K of memory on the Board.

Now with One Year Warranty.

Don't forget the Andromeda 16K RAM Expansion Board $195.00

ANDROMEDA
INCORPORATED
P.O. Box 19144
Greensboro, NC. 27410
919 852-1482

Distributed By

COMPUTER DATA SERVICES
P.O. Box 696
Amherst, NH. 03031
603 673-7375

*Apple II and Applesoft are trademarks

made ("Med. and Dent.") and was highlighted with a short underline. After all changes have been made, the program gives you the option of requesting percentage breakdowns by budget area. This table is shown in Figure 3, which concludes the budgeting routine.

There is nothing very sophisticated about any of this, but the program has been a useful one for our family; there is nothing quite so engaging as seeing an inanimate "object" spring to life at your bidding. And the program was just a lot of fun to write. Let me know of your improvements of it.

Program 1.

```
 10  REM  ************************
 20  REM  *HOME BUDGETING PROGRAM*
 30  REM  *CREATED JUNE 10, 1981 *
 40  REM  *BY WILLIAM R. SWINYARD*
 50  REM  *SHOWS AND CALCULATES  *
 60  RÉM  *DISTRIBUTION OF INCOME*
 70  REM  *INTO BUDGET CATEGORIES*
 80  REM  *INCLUDING REQUESTS FOR*
 90  REM  *UPDATED EXPENDITURES &*
 100  REM *FOR PERCENTAGE DISPLAY*
 110  REM ************************
 115  HOME : PRINT : PRINT : PRINT
      : PRINT : PRINT : PRINT  TAB(
      10)"HOME BUDGETING PROGRAM":
       FOR I = 1 TO 1000: NEXT I: HOME
 120  INPUT "WHAT IS NET MONTHLY I
      NCOME? ";SALARY
 130  PRINT
 232  DIM DD(17),DD$(17)
 236  REM  **BUDGETED ITEMS IN LIS
      TING **
 237  REM  **TO CHANGE BUDGET YOU
      MUST ACCESS THE LISTING DIRE
      CTLY**
 240 DD$(1) = "HOUSING"
 250 DD(1) = 500
 260 DD$(2) = "HEAT"
 270 DD(2) = 50
 280 DD$(3) = "LIGHTS"
 290 DD(3) = 35
 300 DD$(4) = "DONATIONS"
 310 DD(4) = 50
 320 DD$(5) = "TELEPHONE"
 330 DD(5) = 35
 340 DD$(6) = "LOAN PAYMNTS"
 350 DD(6) = 100
 360 DD$(7) = "MED & DENT"
 370 DD(7) = 35
 380 DD$(8) = "GROCERIES"
 390 DD(8) = 300
 400 DD$(9) = "AUTO MAINT."
 410 DD(9) = 25
 420 DD$(10) = "GASOLINE"
 430 DD(10) = 100
 440 DD$(11) = "INSURANCE"
 450 DD(11) = 50
 .
 470 DD$(12) = "CLOTHING"
 480 DD(12) = 50
 490 DD$(13) = "EDUCATION"
 500 DD(13) = 20
 510 DD$(14) = "HOME REPAIR"
 520 DD(14) = 50
 530 DD$(15) = "RECREATION"
 540 DD(15) = 60
 550 DD$(16) = "SAVINGS"
 560 DD(16) = 50
 570 DD$(17) = "OTHER"
 580  REM  ***PRINT OUTPUT***
 590  HOME
 600  PRINT "MONTHLY INCOME",SALARY
 605  PRINT "=============="
 610 SUM = 0
 620  FOR I = 1 TO 17
 630  GOSUB 790: PRINT  TAB( A)I; TAB(
      4)DD$(I); TAB( B)DD(I)
 635  IF I = TT THEN  PRINT  TAB(
      18)"---"
 640 SUM = SUM + DD(I)
 650  NEXT I
 660  PRINT
 670  PRINT "TOTAL BUDGETED =    ";SUM
 680  PRINT "NET AFTER BUDGET = ";
      SALARY - SUM
 685  IF Q = 1 THEN  GOTO 910
 690  FOR X = 1 TO 10000: NEXT X
 700  PRINT "*******************"
 710  REM  ***CHANGES***
 730  INPUT "WAS SPENDING OTHER TH
      AN ABOVE? ";B$
 740  IF B$ = "N" THEN  GOTO 840
 750  INPUT "TYPE ITEM NUMBER, ACT
      UAL EXPEND. ";I,X
 760 TT = I
 770 DD(I) = X
 775  GOTO 600
 780  REM SUBROUTINE FOR DECIMAL PLACEMENT
 790 A = 2:B = 19:C = 25
 800  IF I > 9 THEN A = A - 1
 810  IF DD(I) < 10 THEN B = B + 1
 820  IF DD(I) >  = 100 AND DD(I) <
      1000 THEN B = B - 1
 825  IF DD(I) >  = 1000 THEN B =
      B - 2
 827  IF PC > 9 THEN C = C - 1
 830  RETURN
 835  REM  ROUTINE TO COMPUTE & PRINT %'S
 840  INPUT "WANT %'S OF BUDGET? ";C$
 850  IF C$ = "N" THEN  GOTO 910
 855  PRINT : PRINT
 860  FOR I = 1 TO 17
```

```
865 Q = 0
870 PC = (DD(I) / SUM) * 100
880  GOSUB 790: PRINT  TAB( A)I, TAB(
     4)DD$(I); TAB( B)DD(I), TAB(
     C) INT (PC);"%"
890  NEXT I
895 Q = 1
900  GOTO 660
910  END
```

Figure 1.

```
        HOME BUDGETING PROGRAM
WHAT IS NET MONTHLY INCOME? 1600

MONTHLY INCOME  1600
==============
 1 HOUSING          500
 2 HEAT              50
 3 LIGHTS            35
 4 DONATIONS         50
 5 TELEPHONE         35
 6 LOAN PAYMNTS     100
 7 MED & DENT        35
 8 GROCERIES        300
 9 AUTO MAINT.       25
10 GASOLINE         100
11 INSURANCE         50
12 CLOTHING          50
13 EDUCATION         20
14 HOME REPAIR       50
15 RECREATION        60
16 SAVINGS           50
17 OTHER              0

TOTAL BUDGETED =    1510
NET AFTER BUDGET = 90
```

Figure 2.

```
WAS SPENDING OTHER THAN ABOVE? Y
TYPE ITEM NUMBER, ACTUAL EXPEND. 7,75
MONTHLY INCOME  1600
==============
 1 HOUSING          500
 2 HEAT              50
 3 LIGHTS            35
 4 DONATIONS         50
 5 TELEPHONE         35
 6 LOAN PAYMNTS     100
 7 MED & DENT        75
 8 GROCERIES        300
 9 AUTO MAINT.       25
10 GASOLINE         100
11 INSURANCE         50
12 CLOTHING          50
13 EDUCATION         20
14 HOME REPAIR       50
15 RECREATION        60
16 SAVINGS           50
17 OTHER              0

TOTAL BUDGETED =    1550
NET AFTER BUDGET = 50
```

Figure 3.

```
WAS SPENDING OTHER THAN ABOVE?
WANT %'S OF BUDGET? Y
 1 HOUSING       500    32%
 2 HEAT           50     3%
 3 LIGHTS         35     2%
 4 DONATIONS      50     3%
 5 TELEPHONE      35     2%
 6 LOAN PAYMNTS  100     6%
 7 MED & DENT     75     4%
 8 GROCERIES     300    19%
 9 AUTO MAINT.    25     1%
10 GASOLINE      100     6%
11 INSURANCE      50     3%
12 CLOTHING       50     3%
13 EDUCATION      20     1%
14 HOME REPAIR    50     3%
15 RECREATION     60     3%
16 SAVINGS        50     3%
17 OTHER           0     0%
TOTAL BUDGETED =    1550
NET AFTER BUDGET = 50
```

©

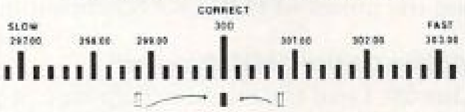
★★★★★★★★★★★★★★★★★★

Bugs in your Apple?

DDT

Disco-Tech's Disc Drive Timer program

zaps disc drive problems!



Analyze disc drive motor speed on a routine basis with an adjustable real-time speedometer. Accurate to one-tenth of one r.p.m. out of 300 r.p.m.

Fine-tune disc drive motor speed yourself. All you need is DDT, two screwdrivers, and five minutes' time.

$29.95 Postpaid
California residents add 6% tax
Diskette & complete manual.

Engineering Business
Architecture Utilities
Surveying

Also available for TRS-80 Model I.

To order or for more information, write or call

disco tech T.M.

microcomputer products a division of

Morton Technologies, Inc.
600 B Street
P.O. Box 11129 • Santa Rosa, CA 95406
707/523-1600
Dealer inquiries invited

★★★★★★★★★★★★★★★★★★

Named GOSUB's

M.R. Smith

In a previous article, (**COMPUTE! #**12), I showed how to use the Ampersand instruction in Applesoft to permit instructions of the form:

```
10 FIRST=1000
20 DEUX=2000
30 THIRD=3000
40 & GOSUB FIRST
50 & GOTO THIRD
```

In Applesoft BASIC, named GOSUB's and GOTO's are not normally allowed.

The machine language program given in that article did not allow the use of names in ON... GOSUB or ON...GOTO statements. The following machine language program rectifies that problem.

The following statements are now permitted:

```
60 FOURTH=4000
70 NUM=INT(1+RND(2))
80 & ON NUM GOSUB FIRST, DEUX
90 & ON NUM GOTO THIRD, FOURTH
```

Using statements of this form makes it much easier to follow programs containing a large number of subroutines. In addition, it is much easier to remember the name of a subroutine rather than its number.

I am presently working on an extension to these statements to allow the passing of variables to a subroutine. That means statements of the form:

```
& GOSUB CALCULATE(A, B, C)
```

allowing the power of FORTRAN subroutine calls.

Description Of The BASIC Program

Line 20. Load the machine language program.

Line 20–100. Demonstrates the & GOSUB and & GOTO statements.

Line 110–180. Demonstrates the & ON... GOSUB and & ON...GOTO statements.

Lines 1000–3500. Demonstration subroutines and statements.

Lines 5000–6160. This subroutine loads and checks the machine language program. Every 17th number is the simple sum of the previous 16 numbers. This allows the entry of the numbers to be checked. The machine language program can be saved using the instruction:

```
BSAVE NAMED.GO,A$300,L$88
```

and called into your programs using the instruction:

```
10 PRINT CHR$(4);"BRUN NAMED.GO"
```

before any call to the ampersand (&) statements are made.

APPLE II*
SOFTWARE FROM
POWERSOFT

PEGASUS

(a PASCAL based data base system)

PEGASUS – is a filing and retrieval system using the PASCAL programming language providing a general means for storing data in an orderly fashion. PASCAL code runs three to five times faster than BASIC code designed for a similar application.

Data stored in the PEGASUS data base may be modified, retrieved, and formatted into convenient reports. Three types of data are supported: character, real, and integer. Each PEGASUS data base record may contain up to 20 fields.

Data may be entered either interactively from the console or as a batch from a text file. Records may be modified after they have been entered or deleted from the data base entirely. PEGASUS may also be used to select groups of records based on the values of one or more fields. Output may be to the CRT screen, a printer or a text file. Thus, PEGASUS may be used to create printed reports, examine data on-line, or *interface with the input or output of other PASCAL programs.* Requirements: Apple II, Plus, or III and two 5¼" disk drives. Or an 8" or Winchester type drive. USCD Pascal Language System.............................
5¼" Disk Only/$199.95

INCOME STATEMENT SYSTEM

INCOME STATEMENT SYSTEM – (Summarized Reports including Budget Figures Based on Super Checkbook III transactions.) – An excellent program complement to SUPER CHECKBOOK III. The system provides for up to 100 income and expense codes. For each code the system maintains a total for the current month, current budget, current year-to-date, and three prior year-to-dates. Income codes may have up to six corresponding expense codes. A "sort code" feature allows account codes to print in a user defined sequence.

Updates to the accounts include current month, end-of-month, and end-of-year. Gross and Net Income Statements may be printed in either account code or sort code sequence. The Account Master File List may be printed by sort code, account code, or alphabetically by account name. Detailed transactions for each code are printed and totaled, one code per page, in code number order.

This system is designed to run in conjunction with the SUPER CHECKBOOK III program described below. Requirements: 48K, two disk drives, printer card, Applesoft
Disk Only/$49.95

SUPER CHECKBOOK III

SUPER CHECKBOOK III – A *vastly improved* version of our popular selling program. With new features such as: simplified but powerful transaction entry and modification routines, new reconciliation routines, additional features such as 30 percent increase in the total number of checks handled, posting of interest from interest bearing checking accounts, automatic teller transactions, bullet proof error handling, and smart disk routines. Plus the program still contains the options of bar graphs, sorting, activities, and account status. See INCOME STATEMENT SYSTEM described above.
Disk Only/Applesoft $49.95

Dealer Inquiries Invited

Visa and MasterCard, Check or Money Order include $2.00 for shipping and handling. C.O.D. $2.00 additional

*Apple II and Applesoft are registered trademarks of APPLE COMPUTER INC

POWERSOFT
P. O. BOX 157
PITMAN, NEW JERSEY 08071
(609) 589-5500

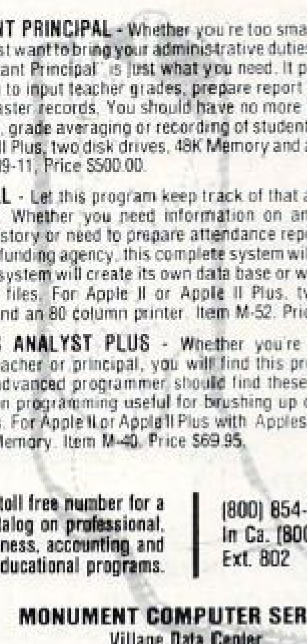
Program 1.

```
 300.387

0300- A9 4C 8D F5 03 A9 10 8D
0308- F6 03 A9 03 8D F7 03 60
0310- C9 B0 F0 0D C9 AB F0 29
0318- C9 B4 F0 31 A2 10 4C 12
0320- D4 20 B1 00 A9 03 20 D6
0320  D3 A5 B9 48 A5 B8 48 A5
0330- 76 48 A5 75 48 A9 B0 48
0338- 20 B7 00 20 44 03 4C D2
0340- D7 20 B1 00 20 7B DD 20
0348- 52 E7 4C 41 D9 20 B1 00
0350- 20 F8 E6 48 C9 B0 F0 0D
0358- C9 AB F0 09 C9 AF D0 BC
0360- 68 20 B1 00 48 C6 A1 D0
0368- 04 68 4C 10 03 A5 A1 48
0370- 20 B1 00 20 7B DD 20 52
0378- E7 20 B7 00 C9 2C D0 05
0380- 68 85 A1 D0 E0 68 68 60
*
```

Program 2.

```
1   REM  M. R. SMITH
2   REM  MAY 1981
3   REM
10   REM  LOAD THE ROUTINE - NORMAL
    GOSUB
20   GOSUB 5000
30   REM  ESTABLISH NAMES OF THE SUBROUTINES
40 FIRST = 1000:DEUX = 1500:THIRD = 2000
50 FOURTH = 2500:FVTH = 3000:SIXTH = 3500
60   REM  DEMONSTRATE NAMED GOSUB'S
70   &  GOSUB FIRST
80   &  GOSUB DEUX
90   REM   DEMONSTRATE   NAMED GOTO
100   &  GOTO FOURTH
110   REM   DEMONSTRATE   NAMED ON...GOSUB
120 NUM =  INT (1 + 3 *  RND (1))
130   &  ON NUM GOSUB FIRST,DEUX,THIRD
140   REM  DEMONSTRATE    NAMED ON...GOTO
150   FOR J = 1 TO 1000: NEXT : REM  DELAY
160 NUM =  INT (1 + 3 *  RND (1))
170   &  ON NUM GOTO FOURTH,FVTH,SIXTH
180   STOP
190   REM
970   REM  DUMMY SUBROUTINES AND LINES
980   REM
990   REM  FIRST SUBROUTINE
1000   PRINT : PRINT "IN SUBROUTINE FIRST": RETURN
1490   REM  SECOND SUBROUTINE
1500   PRINT : PRINT "IN A DIFFERENT NAMED SUBROUTINE"
1510   PRINT "IN SUBROUTINE DEUX": RETURN
1990   REM  THIRD SUBROUTINE
2000   PRINT : PRINT "IN SUBROUTINE THIRD": RETURN
```

CLASS OF '82 . . . BOOT UP!



ASSISTANT PRINCIPAL - Whether you're too small for a big computer or if you just want to bring your administrative duties under your control, the "Assistant Principal" is just what you need. It prints school rosters, allows you to input teacher grades, prepare report cards, and maintain student master records. You should have no more trouble with student scheduling, grade averaging or recording of student grades. For Apple II and Apple II Plus, two disk drives, 48K Memory and an 80 column printer. Item M-3839-11, Price $500.00.

ROLL CALL - Let this program keep track of that all important attendance data. Whether you need information on an individual's tardy, absence history or need to prepare attendance reports for the board or your state funding agency, this complete system will make the task easy work. The system will create its own data base or work off the Assistant Principal's files. For Apple II or Apple II Plus, two disk drives, 48K Memory, and an 80 column printer. Item M-52. Price $250.00.

SYSTEMS ANALYST PLUS - Whether you're a novice computer student, teacher or principal, you will find this program most helpful. Even the advanced programmer should find these seventeen teaching sessions on programming useful for brushing up on the language and commands. For Apple II or Apple II Plus with Applesoft, one disk drive, and 32K Memory. Item M-40. Price $69.95.

Call our toll free number for a free catalog on professional, business, accounting and educational programs.

(800) 854-0561
In Ca. (800) 432-7257
Ext. 802 or write:

MONUMENT COMPUTER SERVICE
Village Data Center
P.O. Box 603
Joshua Tree, California 92252

```
2490  REM  LINE CALLED FOURTH
2500  PRINT : PRINT "LINE CALLED FOURTH": GOTO 120
2990  REM  LINE CALLED FVTH
3000  PRINT : PRINT "LINE CALLED FVTH": GOTO 150
3490  REM  LINE CALLED SIXTH
3500  PRINT : PRINT "LINE CALLED SIXTH": GOTO 150
4980  REM
4990  REM   LOAD IN MACHINE LANGUAGE PROGRAM
5000 LOW = 768:HIGH = 903
5010 OK = 1
5020  REM  LOAD IN GROUP OF SIXTEEN
5030  FOR J = LOW TO HIGH STEP 16
5040 CHECK = 0
5050  FOR K = J TO J + 15
5060  READ IT
5070 CHECK = CHECK + IT
5080  NEXT K
5090  REM   CHECK IF CHECKSUM OKAY
5100  READ SUM
5110 L$ = "OKAY": IF CHECK <  > SUM THEN L$ = "BAD":OK = 0
5120  PRINT L$
5130  NEXT J
5140  IF OK = 0 THEN  STOP
5150  REM  THINGS ARE OKAY - LOAD INTO MEMORY
5160  RESTORE : FOR J = LOW TO HIGH STEP 16
5170  FOR K = J TO J + 15: READ IT: POKE K,IT: NEXT K
5180  READ IT: NEXT J
5190  PRINT "BLOAD OKAY": PRINT : PRINT
5200  REM    SET THE AMPERSAND VECTOR
5210  REM  NOT NEEDED IF CALLED BY BRUN STATEMENT
5220  CALL 768: RETURN
6000  DATA 169,76,141,245,3,169,16,141,246
6010  DATA 3,169,3,141,247,3,96,1868
6020  DATA 201,176,240,13,201,171,240,41,201
6030  DATA 180,240,49,162,16,76,18,2225
6040  DATA 212,32,177,0,169,3,32,214,211
6050  DATA 165,185,72,165,184,72,165,2058
6060  DATA 118,72,165,117,72,169,176,72,32
6070  DATA 183,0,32,68,3,76,210,1565
6080  DATA 215,32,177,0,32,123,221,32,82
6090  DATA 231,76,65,217,32,177,0,1712
6100  DATA 32,248,230,72,201,176,240,13,201
6110  DATA 171,240,9,201,175,208,188,2605
6120  DATA 104,32,177,0,72,198,161,208,4
6130  DATA 104,76,16,3,165,161,72,1553
6140  DATA 32,177,0,32,123,221,32,82,231
6150  DATA 32,183,0,201,44,208,5,1603
6160  DATA 104,133,161,208,224,104,104,96,0,0,0,0,0,0,0,0,1134
```
©

Part II:

A Tape "EXEC" For Applesoft: Loading Machine Language Programs

Sherm Ostrowsky
Goleta, CA

Loading ML With BASIC

This has been an example of the simplest kind of EXEC file; it merely loads and runs a single ML program. Let's extend the procedure a bit more, now, and load both a ML program and an Applesoft program together. One of the most effective advanced programming tools has always been to combine higher level language (e.g., Applesoft) and lower level language (i.e., machine language) routines so that each does the jobs for which it is best suited. But it has long been a problem, much debated and written about in computing magazines, to get them both into the computer from external storage (i.e., cassette tape in this case).

Short ML routines can be included in DATA statements in your Applesoft program and POKEd into memory by a READ - POKE loop. The Lam technique, using strings rather than DATA statements, has also been used to accomplish this. But for ML subroutines of any significant length, this process takes an excruciatingly long time. Some articles have suggested "hiding" long ML routines in front of or behind Applesoft programs by changing certain "pointers" in memory before saving and after loading, to fool Applesoft into believing that they are all parts of one long BASIC program. For one reason or another, none of these techniques has proven very satisfactory.

With the advent of disk systems, this problem was at least solved for disk owners. They just write an EXEC program that loads an Applesoft program and ML subroutines individually stored on the disk. The EXEC program and DOS know where to find them (on the disk) and where to put them (in memory). It is all done automatically. Very simple — *if* you have a disk system. If not ... well, until now you were just sort of out of luck.

Now we cassette users can do the same kind of thing. It takes longer to load, named files are not available, and a little more work is required to arrange everything on the tape and set up the EXEC file (i.e., the loader) to read them and put them where they need to go, but this only has to be done once for any given set of cooperating programs. Thereafter, loading the whole group is just as simple as typing LOAD.

Let me give an example based on a previous article of mine "Clearing the Apple II Low-Resolution Graphics Screen," **COMPUTE!** #10. For those who may not have access to this issue, I'll give all the essential details as they are needed.

Consider the following short Applesoft program that simply calls a subroutine to fill the Low-Resolution screen with a random pattern of colors, asks you to select one of the sixteen low-res color numbers (0–15), then calls a ML subroutine which clears the screen to the selected color in an instantaneous flash. Here is the program:

... load both a ML program and an Applesoft program together.

```
  10 REM FLASH-CLEAR DEMO
  20 GOSUB 1000
  30 VTAB 23
  40 INPUT "SELECT COLOR (0-15):" ;C
  50 COLOR=C
  60 CALL 800
  70 FOR PAUSE = 0 TO 2000: NEXT: GOTO 20
  80 END
1000 GR
1010 FOR I=0 TO 39
1020 FOR J=0 TO 39
1030 COLOR=1 + INT (15 * RND (1))
1040 PLOT J,I
1050 NEXT J,I
1060 RETURN
```

Briefly, line 20 calls subroutine 1000 which fills the screen with random color squares. Line 30 positions the cursor in the text area below the graphics screen. Line 40 asks for your selection and line 50 sets it up to be used in the clearing operation which is performed when line 60 calls the ML subroutine which begins at location (decimal) 800. Line 80 pauses briefly and then loops around to try it all again.

In the original version, the flash-clear subroutine at 800 was stored as DATA within the main program and POKEd into memory using a READ - POKE loop; it's short enough so that this procedure is really quite adequate. But, for the purpose of this demonstration, we will have the Exec-Loader read the ML program in from tape and put it into memory where it is supposed to be. To see how to create the total tape package, you'll have to get the ML program into memory first somehow. The easiest way is to use the Monitor. Do this first, before you enter either the Applesoft loader or the Applesoft demo program. Type CALL - 151, and when you see the asterisk prompt ("*") type:

```
320:A5 30 A0 78 20 2D 03 A0 50 20 3D 03
60 88 99 00 4 99 80 4 99 00 5 99 80 5 D0
F1 60 88 99 00 6 99 80 6 99 00 7 99 80
7 D0 F1 60 (return)
```

Do it exactly as written, with a space between each pair of digits. Just keep right on typing, past the ends of lines, without stopping until you reach the end. The ML program is now in its place in memory. But where we want it to be is on tape.

The order in which things have to go on the tape is this: first the Applesoft loader, then the ML subroutine, and finally the Applesoft demo program. The demo program has to come last because, after the Exec-Loader program (which I will present below) has executed a LOAD for an Applesoft program (here, the demo), it will automatically be deleted out of memory. It is a standard part of an Applesoft LOAD to clear out any pre-existing Applesoft programs first and, although this can be circumvented, there is no reason to go to the trouble of doing so in this case. So the demo must be loaded last because the loader will cease to exist in memory at that time.

This loader is an extension of the one shown before, but with most of the "bells-and-whistles" left off to make it shorter. Aside from changing the memory locations in string Y$ to correspond to the present ML subroutine, the only other change made from the previous loader is the addition of a LOAD command for the demo program.

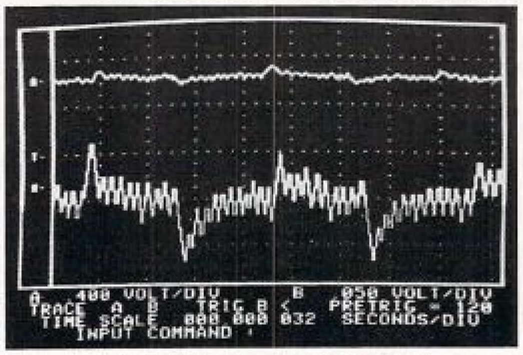
```
10 REM ML + AS LOADER
20 Y$="320.34CR D823G"
30 FOR I=1 TO LEN (Y$): POKE 511+I, ASC
   (MID$ (Y$,I,1)) + 128: NEXT
40 POKE 72,0: CALL - 144
50 POKE 214,85
60 LOAD
70 END
```

This version has no unnecessary features. If there's a loading error, you'll find out about it when the system prints "ERR" on the screen. Since it takes only a short time to load these little programs, there is no need to give you a message so you'll know what is going on during the loading process.

Now here's the procedure, written down in the form of an algorithm: a series of listed steps to be followed one by one.

1. Make sure that the ML subroutine is in memory locations 320 through 34C, as indeed it will be if you typed it in using the Monitor as described above.
2. Go into Applesoft (from the Monitor, type CONTROL-C (return)), and type in the Loader program, as given above.
3. Type: POKE 82,128
4. Start your cassette recorder in RECORD mode and type SAVE (ret). When the loader has been saved, stop the tape, but *do not rewind*.
5. Back to the Monitor (CALL -151). Type:
   320.34CW restart recorder in

DIGITAL STORAGE OSCILLOSCOPE INTERFACES



```
A  .400 VOLT/DIV        B  .050 VOLT/DIV
TRACE  A  B   TRIG B <  PRETRIG = 120
 TIME SCALE  .000 .000 032  SECONDS/DIV
    INPUT COMMAND :
```

APPLESCOPE

Interface for the Apple II Computer

The APPLESCOPE system combines two high speed analog to digital converters and a digital control board with the high resolution graphics capabilities of the Apple II computer to create a digital storage oscilloscope. Signal trace parameters are entered through the keyboard to operational software provided in PROM on the DI control board.

- DC to 3.5 Mhz sample rate with 1024 byte buffer memory
- Pretrigger Viewing up to 1020 Samples
- Programmable Scale Select
- Continuous and Single Sweep Modes
- Single or Dual Channel Trace
- Greater than or less than trigger threshold detection

Price for the two board Applescope system **$595**

***Dealer Inquiries Invited**

Combine an Apple II or S100 based computer system with our interface circuit boards to create a digital storage oscilloscope at a fraction of the cost of other storage scopes.

The S100 interface provides an additional 1024 bytes of buffer memory in place of the PROM. The user must supply the graphics display and driving software. Price of the single board is **$495.**

The SCOPEDRIVER is an advanced software package for the Applescope system. It provides expanded waveform manipulation and digital signal conditioning. The SCOPEDRIVER is available on 5¼" floppy disks for **$49.**

For further information contact:

RC Electronics Inc.
7265 Tuolumne Street
Goleta, CA 93117
(805) 968-6614

Apple Authors:

COMPUTE! is looking for good applications articles on Apple.

RECORD mode, *then* press (return). Stop recorder when finished, but again do not rewind.

6. Back to Applesoft. Type NEW. Type in the demo program. SAVE it in the regular way.

That's it. To check it out, rewind the tape, turn your Apple off and back on (to make sure memory is cleared out), and LOAD the tape in the usual Applesoft manner. You'll hear five beeps before it finishes loading, but, in the end, it stops just the same as any regular Applesoft tape. You run the program by typing RUN.

... we have just written the tape equivalent of an EXEC file which performs the equivalent of a BLOAD.

Does this seem like a lot of trouble to go through just to load a short machine language subroutine along with an Applesoft program? Of course it is. But suppose the ML program were a lot longer. I have a "Print Using" subroutine that is over 1000 bytes long. Have you any idea how many DATA statements it would need to POKE all that into memory from within my calling program, and how long it would take to do it? The fact is that, before the method I have been demonstrating came along, there was no good way at all for a tape user to put long ML subroutines into his Applesoft programs. Only disk users could have that convenience, with their "BLOAD" and "EXEC" commands. Well, we have just written the tape equivalent of an EXEC file which performs the equivalent of a BLOAD. And that's far from all. The possibilities seem endless. I don't want this article to seem the same, so I'll mention just one more idea and then leave you to enjoy cooking up others on your own.

Protecting Programs In Memory

COMPUTE! #11 had an article (page 76) on resolving the memory conflicts between Applesoft programs and the locations of the two hi-res graphics screen memories. The problem is that a really long Applesoft program, starting at its usual location of $0801 (which is decimal 2049) can easily grow to overflow into the first hi-res screen starting at $2000 (decimal 8192), and some can even intrude into the second hi-res screen starting at $4000. An equivalent problem can occur if you want to make use of the second page of lo-res graphics (did you know that there is a second page?) which unfortunately starts at $0800 and thus *always* conflicts with an Applesoft program.

Among the solutions mentioned in the article, the one which seems to me to be the most flexible is to move the whole Applesoft program, variables and all, to a starting location above the memory region you want to protect from interference. For example, if you just want to use page two of lo-res graphics, it would suffice to move the program up so that it begins at $0C01 (decimal 3073). If you want to use both hi-res graphics pages, on the other hand, you'd have to move the program all the way up to $6001 (decimal 24566), assuming you even have that much memory to play around with.

As a matter of fact, you don't actually "move" an Applesoft program around in memory. Although this can be done, it would require doing some repair work on pointers and relinking the program lines which is too technical for most casual users to bother with. It isn't necessary. All you have to do is arrange to have the program go directly into the desired memory locations at the time it is being read in from tape. That is a much simpler procedure, requiring only a couple of POKEs from the keyboard before typing LOAD. Even so, that puts us right back into the situation we were in with ML programs: you can't just type LOAD — you have to refer to a set of written directions in order to get the program to load right. At least you do if your memory is as poor as mine is at recalling such details without notes. And there's another point: if the program is to be usable by a non-computerist, how would you instruct such a person in the loading procedure?

So, here's another job for the Tape-EXEC, alias a loader program. Let the loader do the POKEs for you. That way, once you have gone to a little extra trouble to set the tape up right to begin with, you (or anybody) can forever after just type LOAD and let the computer handle the details. If you think it likely that you might want to use the program more than once or twice, then the extra preparations are worth the trouble in the long run.

A loader for a single Applesoft program, to be entered into memory starting at (say) $0C01, would look something like this:

```
10 REM HIGH MEMORY LOADER
20 POKE 104,12: POKE 3072,0
30 POKE 214,85
40 LOAD
50 END
```

The algorithm for preparing the loader-leader and tape is similar to the one given for the Applesoft/ML conjunction; just omit steps two and five.

If you want the program to load above hi-res page 1, just change line 20 to:

```
20 POKE 104,64: POKE 16384,0
```

If you want it to load above hi-res page 2, change line 20 to:

```
20 POKE 104,96: POKE 24576,0
```

©

Switching Cleanly From Text To Graphics

Brian Nakagawa
Fresno, CA

It is very distracting to watch the Apple II switch from TEXT to GRAPHICS mode or vice versa. Colored squares and grey lines appear briefly when switching from TEXT to LORES graphics while a screen full of inverse characters, usually @ signs, appear when switching from LORES graphics to TEXT. In going from TEXT to HIRES graphics, one sometimes sees the previous HIRES display flash on the screen then dissolve away. In most programs, the extra flash of graphics or characters is merely a distraction. However, to switch cleanly between TEXT and GRAPHICS mode would give the appearance of a more polished program.

This article will show the routines that switch cleanly from TEXT to LORES or HIRES graphics using Applesoft commands and machine level routines already available on the Apple II.

The commands given below can be typed in the immediate mode or run in a program.

TEXT To HIRES Graphics

The simplest way to turn on page 1 of HIRES graphics is to use the command HGR. This first turns the screen to that page then erases the graphics displayed. Type in the following sequence of commands to see this.

```
VTAB 24
HGR
HCOLOR=3
HPLOT 40,140 TO 240,60 TO 40,60 TO 240,140 TO
   140,20, TO 40,140
TEXT
HGR
```

Notice that when you typed in the last HGR the star reappeared briefly then was erased.

Now try these commands. (It is assumed that you are still in HIRES graphics.)

```
HPLOT 40,140 TO 240,60 TO 40,60 TO 240,140 TO
   140,20 TO 40,140
TEXT
POKE 230,32: CALL 62450: HGR
```

Notice that the screen went blank without the brief appearance by the star.

POKE 230,32 sets the HIRES page pointer to page one without changing the current screen display. CALL 62450 is a machine language routine in APPLESOFT that clears the HIRES page indicated by memory location 230. HGR then turns on page one of HIRES graphics.

HIRES page two can be switched to using similar commands. Use the above example and substitute "HGR2" for "HGR" and "POKE 230,64" for "POKE 230,32". Be aware that the text window at the bottom will not be seen.

Switching from HIRES graphics, page one or two, TEXT is easily accomplished typing in the "TEXT" command. If you wish to clear the text page first then use "HOME: TEXT".

TEXT To LORES Graphics

Fill the TEXT screen with text then type in the command "GR". The distracting display of grey lines and colored squares when switching from TEXT to LORES graphics can be avoided with the following commands. Fill the text screen with text before typing in the commands.

```
POKE 230,32: CALL 62450: HGR: CALL -1994: GR
```

If you wish to clear the text window at the bottom of the screen, type in "HOME" or add it to the end of the string of commands as follows:

```
POKE 230,32: CALL 62450: HGR: CALL -1994:
GR: HOME
```

Notice that it is first necessary to clear HIRES page one and go into that page just as was done in the TEXT to HIRES discussion. CALL -1994 changes the text page to inverse @ signs and GR put the screen into LORES page one.

If you are still in LORES graphics type in "TEXT: HOME" which will return you to a clear TEXT page. Did you notice the flash of inverse @ signs? This does not happen all the time so it may be necessary for you to go back into LORES page one with the "GR" command then type in "TEXT: HOME" again.

To avoid this, type in the following commands:

```
GR
HOME: HGR: POKE 34,0: HOME: TEXT
```

GR puts the screen into LORES page one, HOME then clears the text window at the bottom, HGR puts the screen into HIRES page one which is assumed to be clear, POKE 34,0 sets the top of the text window to the top of the screen, HOME clears the text page, and TEXT puts the screen into text mode.

Disadvantages

The code to switch from text to graphics takes more memory and executes slower. You may have noticed the pause when switching between TEXT and GRAPHICS. In a program that switches between TEXT and GRAPHICS often, one can put the switching code in a subroutine to save memory.

The additional code and slightly slower execution speed to switch cleanly is a very small price to pay for the more polished appearance of a program.

References

All of the above commands are documented in the *Applesoft Basic Programming Reference Manual.* Selected commands and the page number they appear on are listed below. POKE 230,32 and POKE 230,64 are documented as general use high resolution graphics locations on page 141. Decimal location 230 is equal to hexadecimal $E6. CALL 62450 and CALL -1994 are on page 134 and POKE 34,0 is on page 29. ©

Interfacing The CCS 7710A Asynchronous Serial Card

Sam Bassett
San Francisco, CA

The following is a list of the connections needed to set up the DT127 to work with the California Computer Systems' 7710A Asynchronous Serial Interface Card for the Apple II:

Pin on the Male DB-25P Connector			**DT127 25 pin Molex**		
Color Pin	**Name**		**Name**	**Pin**	**Recommended**
				1	No contact
3	RD	------------------>	RD	2	Red
6	DSR	------------------>	DSR	3	Green
5	CTS	------------------>	CTS	4	Brown
20	DTR	------------------	RTS	5	Blue
4	RTS	------------------	DTR	6	White
2	TD<	------------------	TD	7	Orange
1,7	GND<	------------------>	GND	8	Black
				9-25	No Contact

The CCS Asynchronous Board is defined as a *D*ata *C*ommunications *E*quipment (DCE) terminal, and the signals at its DB-25S (female) connector are defined in the same way that a modem's would be.

The CCS board transmits its *outgoing* signal on Pin 3 (RD). This must be connected to the DT-127's *incoming* signal connector on the CCS board — Pin 2 (TD).

So far so good — we have information signals being passed back and forth from the printer to the Apple. There is a possible problem, however — the printer can only print so fast, and the Apple can generate and send characters much faster than the printer can print them. Most printers run from 10 to 55 characters per second, which is equivalent to 100 to 550 baud. The Apple can transmit at well over 20,000 baud — 2,000 characters per second. The DT127 has a 625 character buffer built in (expandable to 16K), but if the Apple is sending characters faster than the NEC can print (55 cps), the buffer gets full, characters are lost, and weird things happen to the text that was to be printed.

All is not lost, however — the definition of RS-232 includes several hardware "*handshaking*" signals, and the CCS 7710A (unlike the Apple Inc. Serial and Intelligent Interface Boards) is set up to recognize and use these signals. When the Printer signals that it has enough characters in its buffer, the CCS board will stop sending characters until the printer sends an "OK" signal.

The DT-127 signals that it is OK to send characters to its input on Pin 2 (RD) by making Pin 6 (Data Terminal Ready — DTR) on the Molex connector high — +3 to +12 volts. The CCS board monitors Pin 4 (Request To Send — RTS) to see if the peripheral is ready to receive another character. If Pin 4 goes Low (-3 to -12 volts), the 7710A will *not* send another character until it goes High again.

The CCS 7710A board signals the peripheral that it can accept a character through its Pin 2 (TD) by making its Pin 5 (Clear To Send — CTS) High. The DT-127 watches its Pin 4 (CTS) to see if it is OK to send a message to the computer. If this signal goes Low, it will not send any characters until it goes High again.

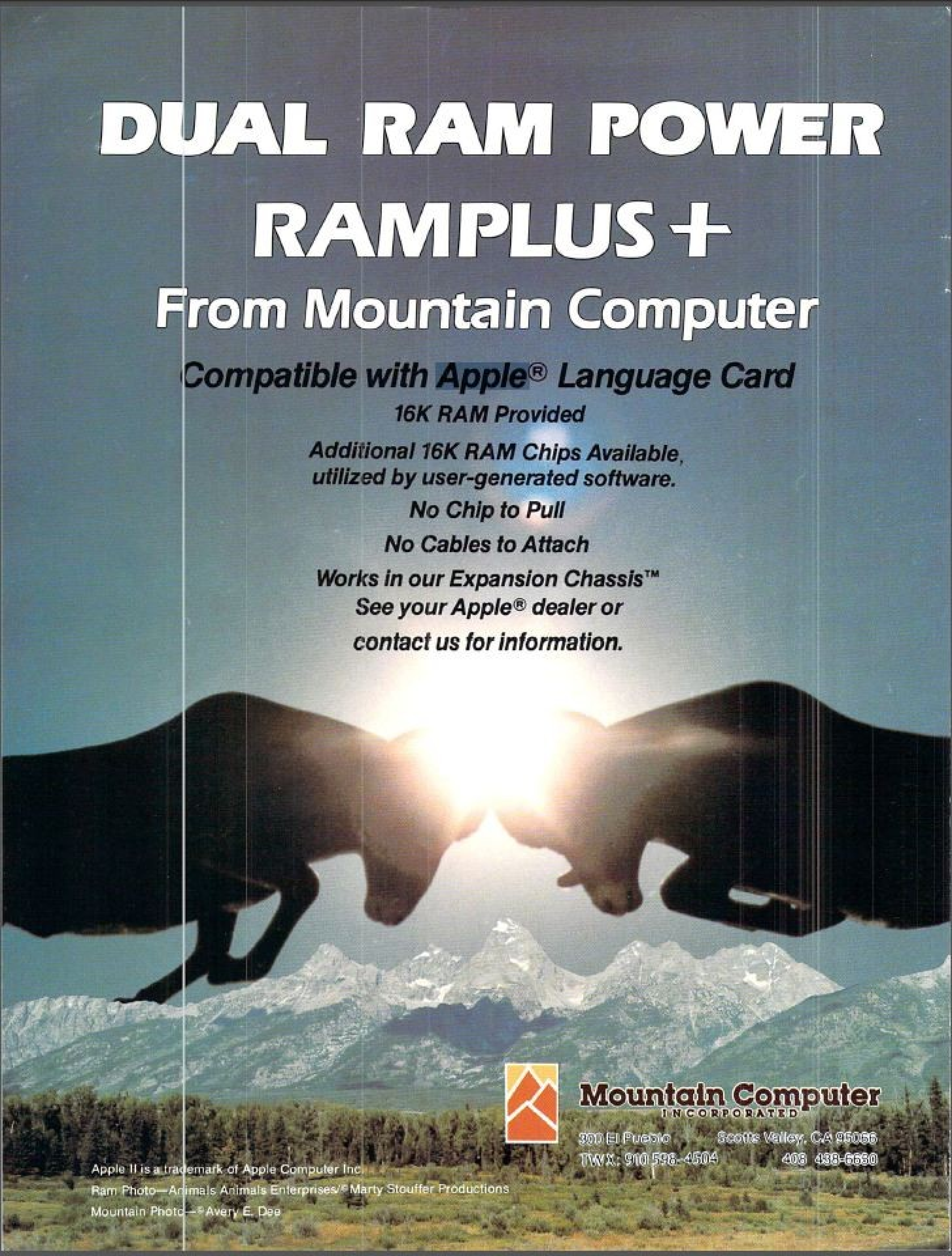
Pins 1 & 7 on the RS-232 connector are connected to Ground, so Pin 8 of the Molex must be connected to one or both of them.

The last two signals are not absolutely necessary, but it is well to hook them up so that nothing is left hanging, or unconnected.

The DT-127's Pin 5 (Request To Send — RTS) is not implemented — it should be High at all times. It should be connected to the CCS board's Pin 20 (Data Terminal Ready — DTR), which tells the Apple: "Yeah, boss, there is somebody out there."

The CCS board's Pin 8 is also permanently at +12 volts, so it should be left unconnected, so that it does not accidentally short to ground and shut down the Apple's power. Pins 9 to 25 (except 20) are not implemented on the CCS board, so they should not be connected to anything.

The CCS board can transmit at any baud rate from 50 to 19,200, and matches the DT-127 perfectly — be sure to check Page 2-1 in the CCS manual for the correct Baud Rate Selector Switch settings, and Page 5 of the DT-127 manual, so that the baud rates being used by both your Apple and your Sellum are the same. ©



DUAL RAM POWER RAMPLUS+

From Mountain Computer

Compatible with Apple® Language Card

16K RAM Provided

Additional 16K RAM Chips Available, utilized by user-generated software.

No Chip to Pull

No Cables to Attach

Works in our Expansion Chassis™

See your Apple® dealer or contact us for information.

Mountain Computer
INCORPORATED

300 El Pueblo Scotts Valley, CA 95066
TWX: 910 598-4504 408 438-6650

Apple II is a trademark of Apple Computer Inc.
Ram Photo—Animals Animals Enterprises/© Marty Stouffer Productions
Mountain Photo—© Avery E. Dee

With the Hayes direct-connect Micromodem II,™ your Apple II can communicate by phone with the outside world. You can access information networks like The Source for a variety of business and personal applications, exchange programs with friends anywhere in North America, and even use your Apple II when you're away from your home or office.

Dependable. The Micromodem II is so dependable it comes with a *two-year* limited warranty. That's another reason why it's the largest selling direct-connect modem for Apple II computers.

Programmable. Automatic dialing and answering? Of course! We include programs on disk that dial phone numbers for you, send messages while you're away, and much more!

Complete. You get everything you need to communicate with other Bell 103 compatible modems at 110 or 300 baud. The serial interface is built-in, and we even include our FCC-approved Microcoupler™ that plugs directly into any modular telephone jack in the U.S.— you don't even need the phone!

S-100, too. The Hayes Micromodem 100 gives S-100 microcomputers all the advantages of our Micromodem II!

Put the outside world *inside* your computer with a data communications system from Hayes. Available at computer stores nationwide—call or write for the location nearest you. And don't settle for anything less than Hayes.

Hayes™

The Hayes Micromodem II opens up your Apple II to the outside world.

Hayes Microcomputer Products Inc. 5835 Peachtree Corners East, Norcross, Georgia 30092 (404) 449-8791

THE apple® GAZETTE

Editor's Note: In February of this year we ran the following Reader's Feedback plea:

> I am a high school science teacher. I am a novice Apple Computer programmer. I would appreciate **COMPUTE!** articles designed to enhance the programming ability of novice Apple programmers... In-depth articles of Apple POKEing, PEEKing, and CALLS would be very helpful....

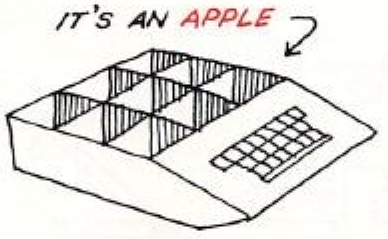
In response, we received the following from Gary who Kathleen and I had the pleasure of meeting at this year's West Coast Computer Faire. Gary, 11, gave us permission to run the response as an article. We think it's an excellent piece for beginners. — RCL

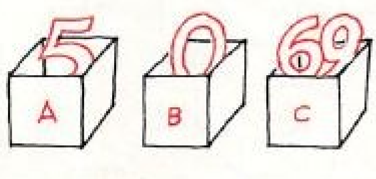
An Apple Primer

Gary Lin
San Jose, CA

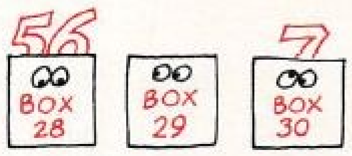
Having trouble with PEEK's, POKE's, and CALL's? Here's a rough explanation:

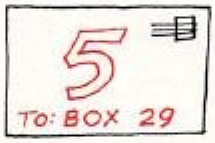
1. Imagine the memory of an Apple is divided up to a bunch of boxes.



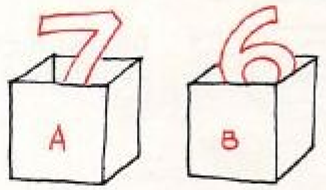


The Apple stores numbers in those boxes. Each box can have only one number assigned to it. Each box has its own personal address.

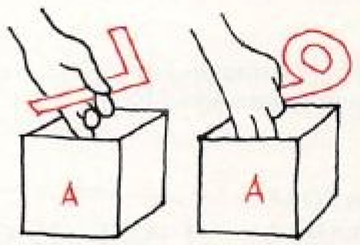




The Apple, like a mailman, gets a number and delivers it to the box. In this example, Box 29 gets the number 5.



The Apple stores numbers in the boxes of its memory. Okay, suppose the Apple sends a 9 to Box A. Box A holds the number 7.



The Apple takes out the 7 and throws it away. Now Box A is clear. ⇨ Then it puts a 9 into Box A.

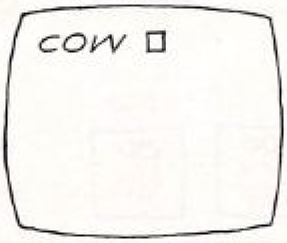
In reality, there are no boxes. Instead there are addresses. Addresses, like boxes, can be assigned a number. For example, address 2 may hold the number 15. The Apple's addresses are numbered in hex, a complicated numbering scheme*. Only the Apple understands HEX, and humans need to know the decimal equivalent.

Don't worry about hex, most beginners say, "What? hex, are you kidding?"

*Base 16

Okay, each address holds a number. If the Apple didn't assign a number to an address, the address automatically holds a 0.

Suppose you type, in BASIC, "COW":



The Apple does a long process and sticks "COW" into its memory. It converts "COW" (or whatever you type in) into little numbers and assigns the numbers to some address. Somewhere, in an address, is "COW."

Well, you can do it a different way!

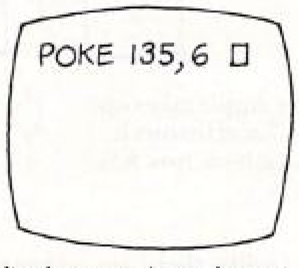
The Secret

Introducing the amazingly, one and only,

POKE!

Okay, POKE is a command that tells the computer to stick a number into an address.

Let's type in POKE 135,6. Here's what the computer does: It converts 6 into hex and runs over to the address. Then 6 is placed in.



```
POKE 135,6 ☐
```

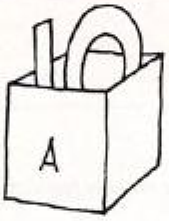
POKE A, B

A is the address (decimal) where you're going to stick a number. B is the number. Try it yourself!

You tell me. What do I do with some number in some address. Here's the next biggie:

PEEK!

Suppose ten is stored someplace, maybe address A. We want to know what is in address A. So we type "PEEK (100)" (100 is the address for A). The computer figures out what 100 is in hex and goes to that address and picks out the number stored there.

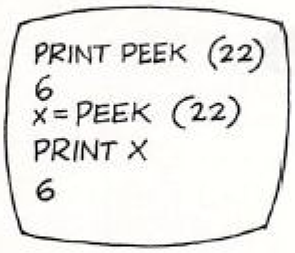


It runs back and converts the number to decimal. To show what is at 100, we PRINT PEEK (100) and it'll print it.

PRINT PEEK (A)

need to show the number at address.

A is the address where you want to know what's there.

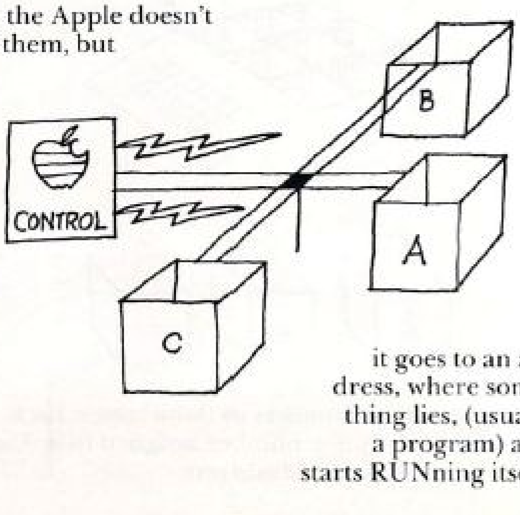


```
PRINT PEEK (22)
6
X=PEEK (22)
PRINT X
6
```

You can assign to a variable (another address, actually) the number which is at location 22 (decimal).

The last, but not least: **CALL!**

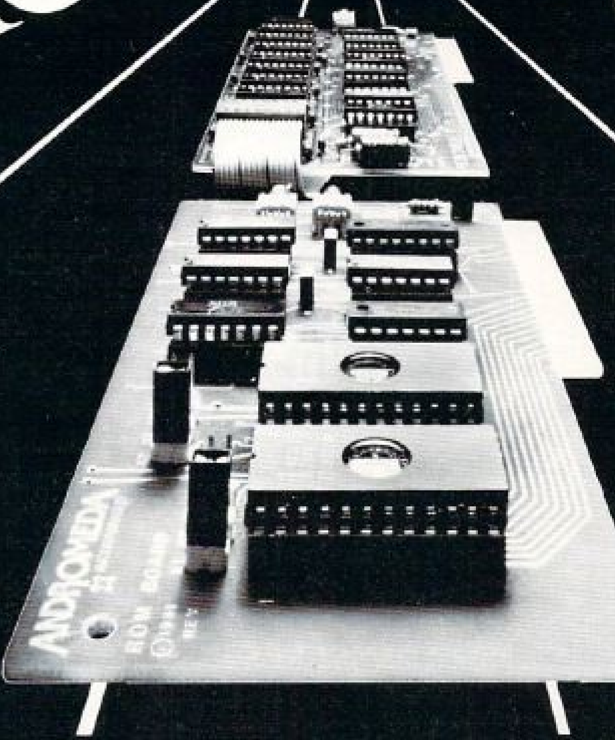
No, the Apple doesn't call them, but



it goes to an address, where something lies, (usually a program) and starts RUNning itself.

A BRIGHT NEW STAR FROM ANDROMEDA!

ROM ✡ RAM



NEW ROM BOARD FOR THE APPLE II*
$125.00 WITH UTILITY ROM.

With Andromeda's new ROM Board, you can plug many useful utility programs into your Apple II. Because ROM memory never forgets, you can access these utilities instantly without having to load them from disk.

The ROM Board comes with the utility ROM, which gives you five powerful options to apply to your Applesoft* programs. With the Utility ROM, you can do automatic line numbering, control a program list with a page mode, restore a crashed Applesoft* program in memory, alphabetize a disk catalogue and create a disk without DOS, giving you an extra 8K on your disk. Any of Soft Control Systems' other ROMS can be used, such as the Dual DOS in ROM, and 'Your'ple ROM.

You can install 2K PROMS, 4K PROMS, or even 2K RAM chips in each of the two memory sockets. So you can even have the Read - Write capability of RAM to develop PROM Programs yourself, or just have an extra 2K RAM for your machine - Language programs. Two 2732 PROMS allow a total of 8K of memory on the Board.

Now with One Year Warranty.

Don't forget the Andromeda 16K RAM Expansion Board $195.00

ANDROMEDA
INCORPORATED
P.O. Box 19144
Greensboro, NC. 27410
919 852-1482

Distributed By:

COMPUTER DATA SERVICES
P.O. Box 696
Amherst, NH. 03031
603 673-7375

*Apple II and Applesoft are trademarks

You say, "that's nice, but what's so big about it?"

The Big Trick of

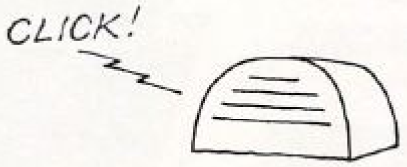
PEEKs, POKEs, and CALLs

Certain addresses in the Apple do nice things, depending on what's stored there.

Like POKE 50,127 (Type it in!)

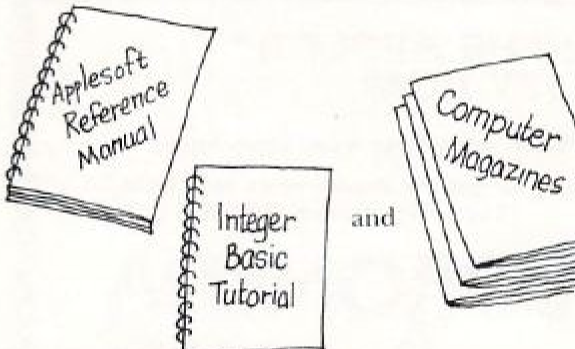
The computer sees 127 is in location 50 and the built-in command tells it to do something. What it does depends on the value stored. Try POKE 50,63 and POKE 50,255. You see, address 50 tells the computer to do something. PEEK does the same, but PEEK doesn't stick a value in — it just activates whatever is at that location.

Like PEEK (-16336)
(Listen carefully!)



CALLs also do stuff like CALL -936 clears the screen. CALL -151 enters Monitor. There are hundreds of POKEs, PEEKs, and CALLs that do special things. That's why people love them.

To find out more, look in these books:



I hope this helps you understand it better, although it's not much of a primer. If you have any questions write me:

Gary Lin
1598 Lock Lomond
San Jose, CA 95129

©

Page Flipper: Five Hires And Four Lores Pages For The Apple

Richard Cornelius
Department of Chemistry
Wichita State University

Five high resolution pages? Four pages of text or low resolution? The facility to copy, overlay or xcopy from one page to another? Yes, all of these and more are available on the (48K) Apple II Plus, and here is the program, PAGE FLIPPER, which demonstrates their use.

Simple arithmetic tells us that space is available on the 48K Apple for more than two high resolution pages. Each hires page occupies 8K (8192 bytes) of memory. Let's digress for a moment to see why that much space is required. The resolution on the Apple is 280 dots across by 192 dots high, which means that the Apple must store information regarding 53760 dots. Each dot is controlled by one bit, so we need $53760 \div 8 = 6720$ bytes to record all of the on/off information for all of the dots on the screen. In addition, we need to record information about the color of the dot.

On the Apple screen, the colors in a horizontal series of seven dots are controlled by a single color control bit. If the color bit is off, then the colors in the seven dots can be any of those given by HCOLOR values of 0 to 3, depending upon the locations of the dots. When HCOLORs 4 to 7 are selected, the color bit is on. The on/off control bits for the seven dots plus the color bit make up one byte. Since each byte controls only seven dots, we need $53760 \div 7 = 7680$ bytes. Some space in the 8K reserved for each page is not used, but there is simply no way to store all of the necessary information in, say, 6K. Even the space on each hires page that is not used to store graphics information cannot readily be used for other purposes because it is fragmented. After every set of 120 bytes that is used for information storage, there follows a set of eight bytes that is not used. Thus 512 bytes of unused space on each high resolution memory page is divided into 64 pieces of eight bytes each.

If we use 8K of memory to store the information on a single hires page, then an Apple with 48K could store enough information for six pages. Since we need to leave some room for a program, we must limit ourselves to five pages. Hires page one is located beginning at 8K and continuing up to 16K, and hires page two occupies memory from 16K to 24K. These two pages are the ones that are readily accessible through the Applesoft commands HGR and HGR2. Pages three, four, and five can be defined to begin at 24, 32, and 40K.

The area in memory to which the HPLOT and DRAW commands write is controlled by a POKE to position 230. POKEing a 32 specifies page one, 64 says page two, and 96, 128, and 160 are used to direct writing to pages three, four, and five. To see that this works, first scroll to the bottom of the Apple screen and then enter the HGR command. Now tell the computer HCOLOR=1 and HPLOT 0,0 to 279, 191 to put a line on the screen. Next POKE 230,64 to direct plotting to hires page two, set HCOLOR=2 and PLOT 279,0 to 0,191. No line appears on the screen because you are viewing page one and the plotting appeared on page two. If you POKE –16299,0 you will switch the Apple to display page two and presto, you see the second line that you plotted.

The Flipper

Unfortunately, we cannot so simply switch to see the other hires pages. Instead we must actually move the information on these pages down to page one or two so that it can be displayed. To accomplish this, we use a short machine language program for speed. This machine language routine is given in Program 1. You don't need to understand any machine language in order to use this little program because it is entered into memory by POKE statements in PAGE FLIPPER, it is executed by a CALL 768, and its function is controlled by POKE statements. Depending upon the POKEd values the routine can a) erase a page, b) copy information from one page to another, discarding the information originally on the destination page, c) overlay a page onto a different one so that the images from the two pages are superimposed, or d) "xcopy" the contents of one page onto another. "Xcopy" is most easily described as being analogous to the XDRAW routine which handles shapes in Applesoft. If you XDRAW a shape on top of some existing image, you get a composite of both the first image and the shape. If you XDRAW again, the shape disappears and you are left with only the original image. In PAGE FLIPPER, if you "xcopy" the contents of one page onto another and then xcopy it again, you are left with the original image also. For those interested in the machine language, "xcopy" uses an exclusive or (EOR) while overlay uses an inclusive or (ORA).

PAGE FLIPPER can also manipulate the pages of memory which store text or low resolution (lores) graphics. Just as the Apple has two hires pages, it also has two text/lores pages which begin at 1K and 2K. Much less information is required to store the letters that appear on the screen than is needed to store a screenful of hires graphics, so each text/hires page occupies only 1K of memory. Since the text screen offers 40 characters across and 24 down, there are 960 "boxes" where characters can be displayed. The contents of each little box is controlled by one byte of memory, so 960 bytes are all that is needed.

As is the case for the hires screens, the unused memory within the 1K allocated for a text/lores page is fragmented into many 8-byte pieces. The image on the lores screen corresponds to the same information as the text screen, but the image displayed is different when the Apple is in lores mode. Each byte which specifies a character on the text screen determines the colors (COLOR 0 to 15) of two blocks on the lores screen which occupy the same screen location as the corresponding character. Four bits (a nibble) determine the color of the upper block, and four bits determine the color of the lower block. In PAGE FLIPPER page three of text/lores is at 3K and page four is at 4K. Page four is used only to save the instructions. A schematic map of memory usage in the program is given in Figure 1. When the machine language routine is used to move any of the text/lores pages, it has less to move than when it operates on any of the hires pages, so another POKE statement is used to specify the size of the page that is being moved.

In addition to the POKEs used to adapt the machine language routine to different purposes, POKE statements are also used to control the display mode of the Apple. These POKEs are outlined in the Apple manuals. All of the POKE positions used in PAGE FLIPPER are listed in Table 1.

Easily Moved Pages

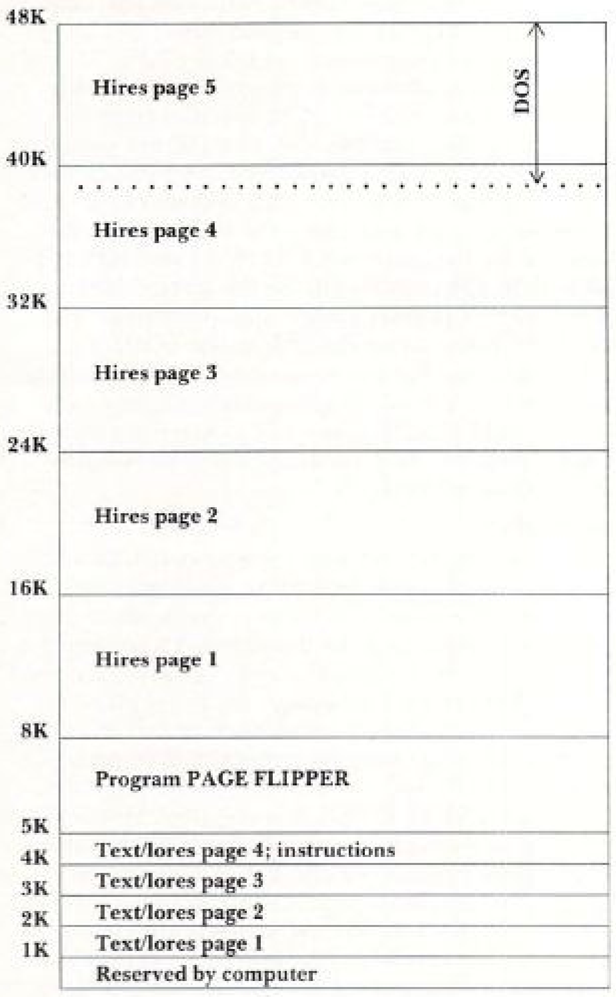
Now that the memory layout and details have been explained, let's look at the PAGE FLIPPER program itself. The first thing to notice is that the program is divided into two parts. The initial part completes a few tasks and then loads the second part. This division is necessary for two reasons. One reason is that the division leaves that part of the program which does most of the work (the second part) small enough that room is left for three pages of text/lores memory plus a fourth page for the directions. The other, more critical, reason is that the first program POKEs certain values into the correct positions so that the second program loads above text/lores page four. Normally an Applesoft program loads starting at 2K, but we want to be able to copy images into that area without overwriting our program.

The first executable statements in the initial part of the program POKE into memory the machine language transfer routine so that it will be there when it is needed. Beginning in line 2000, the instructions are printed, but they are printed on text page one while hires graphics page two is displayed so the user doesn't see them yet. In line 2190, these instructions are moved into the memory area for hires page one for safekeeping while the second half of the program is loaded later. After

Table 1. POKE Positions and Functions

Positions	Values to be POKEd	Function
103,104	1,20	sets spot for beginning of program to above text/lores page 4
230	32,64,96,128, or 160	makes HPLOT write on hires page 1, 2, 3, 4, or 5
768 to 804	values in statement number 1040	puts machine language transfer routine into memory
773	32 x hires page no. or 4 x lores page no.	determines page from which image will be taken
781	32 x hires page no. or 4 x lores page no.	determines page to which image will be written
785	32 for hires, 4 for text/lores	sets size of page to be transferred
790,791	169,0	erase page [LDA #$00]
	177,6	copy a page [LDA ($06),Y]
	17,8	overlay [ORA ($08),Y]
	81,8	xcopy [EOR ($08),Y]
5120,5121,5122	0,0,0	allows execution of program loaded above text/lores page 4
-16297	0	display hires if in graphics mode
-16298	0	display lores if in graphics mode
-16299	0	display page 1 (hires or text/lores)
-16300	0	display page 2 (hires or text/lores)
-16301	0	mix text and graphics if in graphics mode
-16302	0	full screen graphics if in graphics mode
-16303	0	text mode
-16304	0	graphics mode

Figure 1. Schematic RAM Memory Map



this information is moved, text page one is cleared and the introductory screen image is printed. The next to last action in the initial part is moving the pointers which specify where Applesoft programs begin so that when the last statement runs the second part of the program, it will load above text/lores page four.

For convenience in reference, the statements in the second part of the program are numbered higher than those in the first part, but this numbering scheme is not required for its operation. As part of the initialization routine, HIMEM is set to 8192 in order to prevent string variables from being written onto one of the hires pages. The commands IN#0 and PR#0 disconnect DOS so that DOS can be erased by using the machine language transfer routine. Until DOS is disabled hires pages four and five cannot be used. The remainder of the initialization routine plots random lines on the hires pages two through five in different colors, copies the instructions from hires page one to text page one, and then clears and plots random lines on hires page one.

When the program reaches the input routines, we can see the power of being able to readily move pages in memory. The directions that are displayed on the screen are given in Figure 3 except that the inverse characters on the screen are represented only by underlines in the figure. All of the input is controlled by GET statements so that the user never needs to hit return. At any time, an "I" will move the instructions to text page one and display them. A "T" puts the display into text mode, "L" gives lores graphics, and an "H" changes to hires graphics. The commands "M" and "F" specify mixed and full screen graphics. "Q" is used to quit the program. The "Q" reboots DOS which was cleared to make room for hires pages four and five. The command "D" followed by a one or two shows page one or two which may be text, lores, or hires depending on which keys have been pressed previously. An "E" followed by a number can erase any one of the hires pages 1-5 or text/lores pages 1-3 depending upon the current mode of display

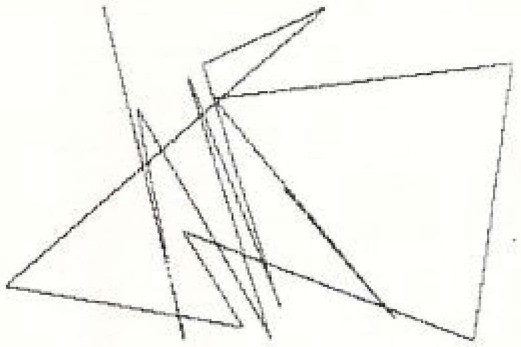
Figure 2. Xcopying with Page Flipper

A) Random lines on page 2.

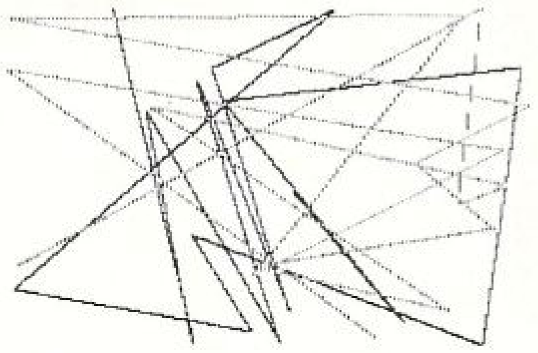
B) Page 3 xcopied onto page 2.

C) Page 3 xcopied onto page 2 again.

A)



B)



C)

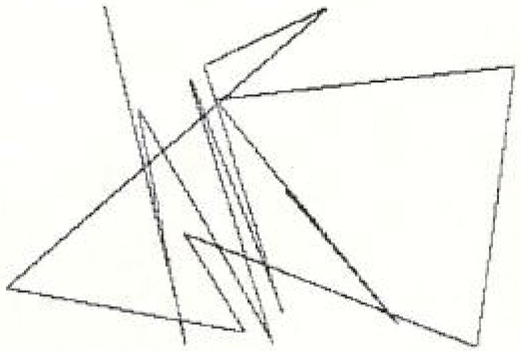


Figure 3. Instruction Page

Options: Instructions

TEXT, LOW — OR HIGH-RES GRAPHICS

MIXED OR FULL SCREEN GRAPHICS

QUIT AND REBOOT

The following commands must be followed by one or two page numbers (represented by X and Y). Accessible pages are Hires 1-5 (1-2 for display) and Text/Lores 1-3 (1-2 for display).

DISPLAY X

ERASE X

COPY X ONTO Y

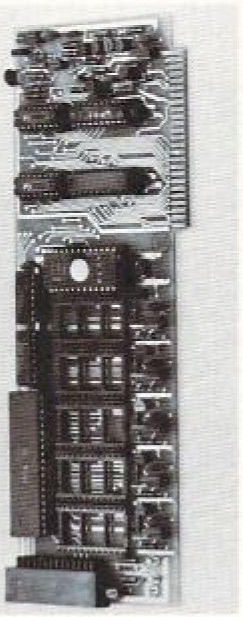
XCOPY X ONTO Y

OVERLAY X ONTO Y

OMNI

The "Everything" I/O Board for the Apple II & II+

OMNI is a multi-function input/output board for the Apple II or II+ computer. It provides, on a single board, most of the "missing" features needed to make the Apple a complete computer. With OMNI your Apple can have:



- Parallel I/O with handshaking
- RS232 Level serial I/O (software driven)
- 24 Hr. Real Time Clock with Alarm
- 2K EPROM with graphics, I/O driver, and screen editor firmware
- 256 Byte PROM supervisory firmware
- Six 2K PROM/EPROM expansion sockets spftware selectable (one socket used for Eclectic firmware, 5 available for user)

INPUT:
- Generate full ASCII character set from keyboard
- Optional shift key detection
- User-definable "soft" keys with screen legends
- Integrated text line editor full cursor movement, insertion/deletion modes
- Key legend stickers included
- Demonstration Diskette with programming examples and a Soft Character Editor

OUTPUT:
- Full 96 character ASCII display
- Concurrent 64 user-definable "soft" characters
- Optional character overstrike and EOR on background
- Optional double-width color characters
- Character rotation in 90° steps

Never before have so many functions been available on a single board.

OMNI was designed with one major goal in mind, flexibility. The OMNI system consists of some extremely simple but very sophisticated hardware, a large amount of powerful firmware (programs permanently residing in Read Only Memory chips), and an equally extensive amount of software (programs residing on diskette that are loaded into RAM as needed). In addition, OMNI comes with extensive documentation.

ECLECTIC SYSTEMS CORPORATION

To order TOLL FREE: 1-800- 527-3135
or Order by Mail Below

Eclectic Systems Corporation, P.O. Box 1166, Addison, TX 75001

Here's my order for OMNI at $268 plus $3 for shipping and handling (UPS surface, unless specified), $5 overseas.

☐ My check is enclosed.

☐ Please charge my VISA ____ Mastercard ____

Account # __________ Expires __________

Name ______________________

Address ______________________

City/State __________ Zip __________

Signature ______________________

(hires or text/lores). The commands "C", "X", and "O" each need to be followed by two numbers. The first number gives the source page and the second number the destination page for the transfer routine.

As an example of what PAGE FLIPPER can demonstrate, press the following sequence of letters, observing what (if anything) happens after each entry; L, H, D, 2, X, 3, 2, X, 3, 2. This sequence changes the display first to lores page one, then the white random lines of hires page one, and then to the green random lines of hires page two (Figure 3a). The X command xcopies the violet lines of hires page three onto two to give a result like in Figure 3b and then again xcopies three onto two. The result is exactly the image that was first on page two. Next try to xcopy first the green lines of page two, then the violet lines of page three, the orange lines of page four, and the blue lines of page five onto page one: D, 1, X, 2, 1, X, 3, 1, X, 4, 1, X, 5, 1. Now that all of these lines are on page one, xdraw pages two through five onto page one once again. Color by color, all of the lines disappear except the white ones that were originally on page one. Now try your hand with the lores screens in the same manner. Remember that at any time you can press "I" to return to the instruction page.

This program by itself is fun to play with, but does not actually accomplish much. The accomplishment will come when you make use of the concepts pulled together here and put them in your own programs to improve them. Imagine the enhanced graphics capabilities that you will have with five hires pages. Get to work!

Program 1: Machine Language Transfer Routine

Location	Value	Operation	
0300-	A9 00	LDA	#$00
0302-	85 06	STA	$06
0304-	A9 20	LDA	#$20
0306-	85 07	STA	$07
0308-	A9 00	LDA	#$00
030A-	85 08	STA	$08
030C-	A9 20	LDA	#$20
030E-	85 09	STA	$09
0310-	A2 20	LDX	#$20
0312-	A0 00	LDY	#$00
0314-	B1 06	LDA	($06),Y
0316-	A9 00	LDA	#$00
0318-	91 08	STA	($08),Y
031A-	88	DEY	
031B-	D0 F7	BNE	$0314
031D-	E6 07	INC	$07
031F-	E6 09	INC	$09
0321-	CA	DEX	
0322-	D0 F0	BNE	$0314
0324-	60	RTS	

Program 2: "Page Flipper"

```
100  REM *** PAGE FLIPPER ***
110  REM     INITIAL PART

130  REM  BY DICK CORNELIUS
140  REM     CHEMISTRY DEPT.
150  REM     WICHITA STATE UNIV.
160  REM     WICHITA, KS  67208

170  REM     (316) 689-3120

180  REM  POKES USED BY THE MACHINE LANGUAGE TRANSFER ROUTINE:
190  REM   773:  SPECIFIES THE "FROM" PAGE
200  REM         32,64,96,128, OR 160 FOR HIRES PAGE 1,2,3,4, OR 5
210  REM         4,8,12, OR 16 FOR TEXT/LORES PAGE 1,2,3, OR 4
220  REM   781:  SPECIFIES THE "TO" PAGE (SEE VALUES ABOVE)
230  REM   785:  SIZE OF PAGE:  4-TEXT/LORES, 8-HIRES
240  REM   790:  DETERMINES (WITH 791) THE NATURE OF THE TRANSFER
250  REM         177-COPY, 81-XCOPY, 17-OVERLAY, 169-ERASE
260  REM   791:  0-ERASE, 6-COPY, 8-OVERLAY OR XCOPY

1000  REM
        **INITIALIZATION**

1010  REM    THE NEXT TWO STATEMENTS POKE INTO MEMORY THE MACHINE
1020  REM    LANGUAGE MEMORY TRANSFER ROUTINE
1030  FOR SPOT = 768 TO 804: READ CODE: POKE SPOT,CODE: NEXT
1040  DATA  169,0,133,6,169,32,133,7,169,0,133,8,169,64,133,9,162,32,
     160,0,177,6,81,8,145,8,136,208,247,230,7,230,9,202,208,240,96
2000  REM
        **PRINT INSTRUCTIONS**
```

```
2010  HIMEM: 8192
2020  HOME : HGR2
2030  HOME
2040  PRINT "OPTIONS:";
2050  HTAB 10: INVERSE : PRINT "I";: NORMAL : PRINT "NSTRUCTIONS"
2060  PRINT : HTAB 10: INVERSE : PRINT "T";: NORMAL : PRINT "EXT, ";:
      INVERSE : PRINT "L";: NORMAL : PRINT "OW- OR ";: INVERSE : PRINT
     "H";: NORMAL : PRINT "IGH-RES GRAPHICS";
2070  PRINT : HTAB 10: INVERSE : PRINT "M";: NORMAL : PRINT "IXED OR
     ";: INVERSE : PRINT "F";: NORMAL : PRINT "ULL SCREEN GRAPHICS"
2080  PRINT : HTAB 10: INVERSE : PRINT "Q";: NORMAL : PRINT "UIT AND
     REBOOT"
2090  PRINT : PRINT "THE FOLLOWING COMMANDS MUST BE FOLLOWED"
2100  PRINT "BY ONE OR TWO PAGE NUMBERS (REPRESENTED"
2110  PRINT "BY X AND Y). ACCESSIBLE PAGES ARE HIRES"
2120  PRINT "1-5 (1-2 FOR DISPLAY) AND TEXT/LORES 1-3";
2130  PRINT "(1-2 FOR DISPLAY).
2140  PRINT : HTAB 10: INVERSE : PRINT "D";: NORMAL : PRINT "ISPLAY "
     ;: INVERSE : PRINT "X": NORMAL
2150  PRINT : HTAB 10: INVERSE : PRINT "E";: NORMAL : PRINT "RASE ";:
      INVERSE : PRINT "X": NORMAL
2160  PRINT : HTAB 10: INVERSE : PRINT "C";: NORMAL : PRINT "OPY ";: INVERS
E : PRINT "X";: NORMAL : PRINT " ONTO ";: INVERSE : PRINT "Y": NORMAL
2170  PRINT : HTAB 10: INVERSE : PRINT "X";: NORMAL : PRINT "COPY ";:
      INVERSE : PRINT "X";: NORMAL : PRINT " ONTO ";: INVERSE : PRINT
     "Y": NORMAL
2180  PRINT : HTAB 10: INVERSE : PRINT "O";: NORMAL : PRINT "VERLAY "
     ;: INVERSE : PRINT "X";: NORMAL : PRINT " ONTO ";: INVERSE : PRINT
     "Y": NORMAL
2190  POKE 773,4: POKE 781,32: POKE 785,4: POKE 790,177: POKE 791,6: CALL
     768: REM MOVES PAGE 1 TO HIRES PAGE 1
3000  REM
        **PRINT FIRST SCREEN IMAGE**

3010  TEXT : HOME : VTAB 2: HTAB 13:
      PRINT "'PAGE FLIPPER'
3020  PRINT : PRINT : PRINT "THIS PRO
      GRAM ALLOWS THE EASY DISPLAY OF"
3030  PRINT "THE VARIOUS TEXT AND
      GRAPHICS PAGES AND"
3040  PRINT "DEMONSTRATES THE 'XPLOT'
      UTILITY."
3050  PRINT : PRINT "WRITTEN BY DICK
      CORNELIUS
3060  PRINT " WICHITA STATE UNIVERSITY"
3070  PRINT " WICHITA, KS 67208"
3080  PRINT : PRINT : PRINT : PRINT
      "PLEASE WAIT A FEW SECONDS..."
3090  VTAB 21
4000  REM
        **LOAD SECOND HALF**

4010  POKE 104,20: POKE 103,1: REM MOVES
      STARTING POSITION FOR APPLESOFT
      PROGRAMS
4020  POKE 5120,0: POKE 5121,0: POKE
      5122,0: REM  PUTS ZEROS INTO
      STARTING POSITIONS
4030 D$ =  CHR$ (13) +  CHR$ (4)
4040  PRINT D$"RUN PAGE FLIPPER.FINAL
      PART"
```

★★★★★★★★★★★★★★★★★

Bugs in your Apple?

DDT

Disco-Tech's Disc Drive Timer program

zaps disc drive problems!

CORRECT
300
SLOW
297.00
298.00
299.00
301.00
302.00
FAST
303.00

Analyze disc drive motor speed on a routine basis with an adjustable real-time speedometer. Accurate to one-tenth of one r.p.m. out of 300 r.p.m.

Fine-tune disc drive motor speed yourself. All you need is DDT, two screwdrivers, and five minutes' time.

$29.95 Postpaid
California residents add 6% tax
Diskette & complete manual.

Engineering Business
Architecture Utilities
Surveying

Also available for TRS-80 Model I.

To order or for more information, write or call

disco tech T.M.

microcomputer products
a division of
Morton Technologies, Inc.
600 B Street
P.O. Box 11129 • Santa Rosa, CA 95406
707/523-1600
Dealer inquiries invited

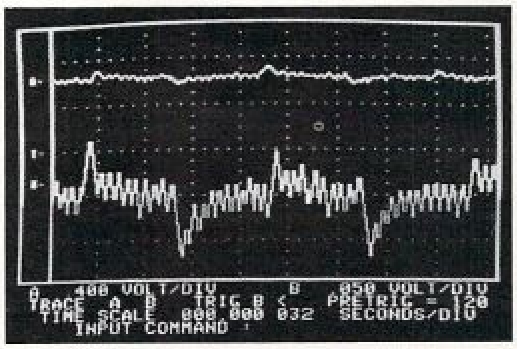
★★★★★★★★★★★★★★★★★

```
5000  REM ***PAGE FLIPPER***
5010  REM     FINAL PART
5020  REM   UPDATED 6/22/81

5030  REM  BY DICK CORNELIUS
5040  REM     CHEMISTRY DEPT.
5050  REM     WICHITA STATE UNIV.
5060  REM     WICHITA, KS  67208

5070  REM     (316) 689-3120

6000  REM
        **INITIALIZATION**

6010  HIMEM: 8192: REM  KEEPS ALL VARIABLES STORED BELOW HIRES PAGE 1
6020 BELL$ =  CHR$ (7):MODE$ = "T/L"
6030  IN# 0: REM    THESE TWO COMMANDS
6040  PR# 0: REM    DISCONNECT DOS
6050  REM  THE FOLLOWING STATEMENTS CLEAR THE VARIOUS PAGES
6060  POKE 790,169: POKE 791,0
6070  POKE 785,128: POKE 773,32: POKE 781,64: CALL 768: REM CLEARS HIRE
     S PAGES 2-5
6080  REM DOS HAS NOW BEEN ERASED
6090  REM
        **DRAWING LINES ON DIFFERENT PAGES**

6100  FOR PAGE = 2 TO 5
6110  HCOLOR= PAGE - 1: IF PAGE > 3 THEN  HCOLOR= PAGE + 1
6120  GOSUB 8000: REM LINES PLOTTED ON HIRES PAGES 2-5 HERE
6130  NEXT
6140  POKE 773,32: POKE 781,4: POKE 785,4: POKE 790,177: POKE 791,6: CALL
     768: REM MOVES INSTRUCTIONS ONTO PAGE 1
6150  POKE 781,16: CALL 768: REM MOVES INSTRUCTIONS ONTO PAGE 4
6160  POKE 781,32: POKE 785,32: POKE 790,169: POKE 791,0: CALL 768: REM
     CLEARS HIRES PAGE1
6170 PAGE = 1: HCOLOR= 3: GOSUB 8000: REM LINES PLOTTED ON HIRES PAGE 1
7000  REM
        **INPUT ROUTINES**

7010  GET G$
7020  IF G$ = "I" THEN  PRINT BELL$;: GOSUB 7900
7030  IF G$ = "T" THEN  PRINT BELL$;: POKE  - 16303,0:MODE$ = "T/L"
7040  IF G$ = "L" THEN  PRINT BELL$;: POKE  - 16298,0: POKE  - 16304,0:
     MODE$ = "T/L"
7050  IF G$ = "H" THEN  PRINT BELL$;: POKE  - 16297,0: POKE  - 16304,0:
     MODE$ = "H"
7060  IF G$ = "M" THEN  PRINT BELL$;: POKE  - 16301,0
7070  IF G$ = "F" THEN  PRINT BELL$;: POKE  - 16302,0
7080  IF G$ = "Q" THEN 9000
7090  IF G$ = "D" THEN  PRINT BELL$;: GOSUB 7400
7100  IF G$ = "E" THEN  PRINT BELL$;: GOSUB 7500
7110  IF G$ = "C" OR G$ = "X" OR G$ = "O" THEN  PRINT BELL$;: GOSUB 760
     0
7120  GOTO 7010
7400  REM
        **DISPLAY**

7410  GET G$
7420  IF G$ = "1" THEN  POKE  - 16300,0: PRINT BELL$;: RETURN
7430  IF G$ = "2" THEN  POKE  - 16299,0: PRINT BELL$;: RETURN
7440  POP : GOTO 7030
7500  REM
```

```
          **ERASE**

7510  GOSUB 7800
7520  POKE 781,PLOC
7530  POKE 790,169:POKE 791,0
7540 SIZE = 32
7550  IF MODE$ = "T/L" THEN
      SIZE = 4
   **COPY, XCOPY, OR OVERLAY**
7560  POKE 785,SIZE
7570  CALL 768
7580  RETURN
7600  REM

7610 SIZE = 32
7620  IF MODE$ = "T/L" THEN
      SIZE = 4
7630  POKE 785,SIZE
7640 A1 = 177:A2 = 6
7650  IF G$ = "X" THEN A1 =
      81:A2 = 8
7660  IF G$ = "O" THEN A1 =
      17:A2 = 8
7670  GOSUB 7800
7680  POKE 773,PLOC
7690  GOSUB 7800
7700  POKE 781,PLOC
7710  POKE 790,A1
7720  POKE 791,A2
7730  CALL 768
7740  RETURN
7800  REM
          **PAGE SELECTOR**

7810  GET G$
7820 PAGE =  ASC (G$) - 48
7830 MAX = 5:MULT = 32
7840  IF MODE$ = "T" OR MODE$ = "T/L" THEN MAX = 3:MULT = 4
7850  IF PAGE < 1 OR PAGE > MAX THEN  POP : POP : GOTO 7030
7860 PLOC = PAGE * MULT: REM  PLOC IS PAGE LOCATION
7870  PRINT BELL$;: RETURN
7900  REM
          **GET INSTRUCTIONS**

7910 MODE$ = "T/L"
7920  POKE 773,16: POKE 781,4: POKE 785,4: POKE 790,177: POKE 791,6: CALL
     768
7930  POKE  - 16303,0: POKE  - 16300,0
7940  RETURN
8000  REM
          **PLOT RANDOM LINES**

8010  POKE 230,PAGE * 32
8020  HPLOT 280 *  RND (1),192 *  RND (1)
8030  FOR LINE = 1 TO 15
8040 X = 280 *  RND (1):Y = 192 *  RND (1)
8050  HPLOT  TO X,Y
8060  NEXT
8070  RETURN
9000  REM
          **REBOOT**

9010  PR# 6
```

©

DIGITAL STORAGE OSCILLOSCOPE INTERFACES



```
A  400 VOLT/DIV      B   050 VOLT/DIV
TRACE  A  B   TRIG B <   PRETRIG = 120
 TIME SCALE  000.000 032  SECONDS/DIV
    INPUT COMMAND :
```

APPLESCOPE

Interface for the Apple II Computer

The APPLESCOPE system combines two high speed analog to digital converters and a digital control board with the high resolution graphics capabilities of the Apple II computer to create a digital storage oscilloscope. Signal trace parameters are entered through the keyboard to operational software provided in PROM on the DI control board.

- DC to 3.5 Mhz sample rate with 1024 byte buffer memory
- Pretrigger Viewing up to 1020 Samples
- Programmable Scale Select
- Continuous and Single Sweep Modes
- Single or Dual Channel Trace
- Greater that or less than trigger threshold detection

Price for the two board Applescope system **$595**

*Dealer Inquiries Invited

Combine an Apple II or S100 based computer system with our interface circuit boards to create a digital storage oscilloscope at a fraction of the cost of other storage scopes.

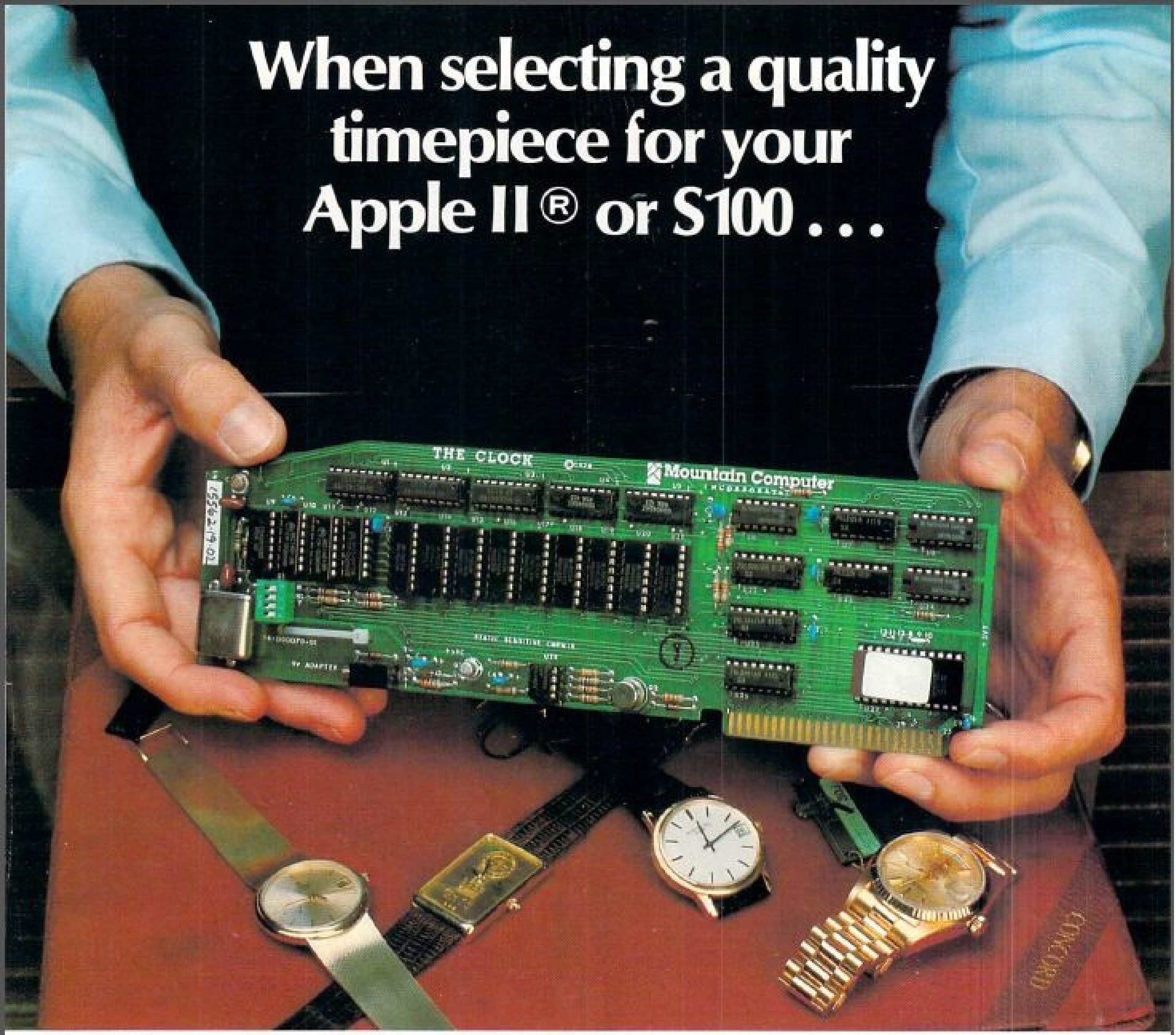
The S100 interface provides an additional 1024 bytes of buffer memory in place of the PROM. The user must supply the graphics display and driving software. Price of the single board is **$495.**

The SCOPEDRIVER is an advanced software package for the Applescope system. It provides expanded waveform manipulation and digital signal conditioning. The SCOPEDRIVER is available on 5¼" floppy disks for **$49.**

For further information contact:

RC Electronics Inc.
7265 Tuolumne Street
Goleta, CA 93117
(805) 968-6614

When selecting a quality timepiece for your Apple II® or S100 . . .



Select The Clock™ or 100,000 Day Clock™ from Mountain Computer.

100,000 DAY CLOCK

- Utilizes 16 I/O ports of a Z-80 or 8080 system.
- Crystal controlled circuit for accurate time: ± .001%.
- Time in 100mS increments for periods up to 100,000 days (273 years).
- Software selectable interrupts for intervals of 100mS, 1mS, . . . 1 sec., 10 sec, 1 min, 1 day, 10 days, . . . etc.

THE CLOCK

- Crystal controlled circuit for accurate time: ± .001%.
- Software and hardware selectable interrupts.
- The standard for software compatibility.

See your Apple® Dealer or contact us for more information.

Mountain Computer
INCORPORATED

300 El Pueblo, Scotts Valley, CA 95066
TWX: 910 598-4504 (408)438-6650

The word "Mountain" and the logo are trademarks of Mountain Computer, Inc.

Part Two:

An Introduction To Binary Numbers

Charles Brannon
Greensboro, NC

This is the second in a series of articles on elementary computer arithmetic. The previous article, Part One, described the binary numbering system, as used on a microcomputer. We will now delve into the use of binary numbers — adding and subtracting.

We'll start with the simplest one first — addition. Besides, you have to know how to add before you can subtract. As you might have realized, binary addition should be rather simple, since you are only adding ones and zeros. The few complications involve the *carry*. Just to refresh you on that, here is a sample base ten addition:

```
  23
+ 51
  ??
```

To add 23 and 51, we add the numbers digit by digit, from right to left. So first we add 3 and 1 to get 4, which we place underneath the digits added. Next we add the 2 and the 5, and place a 7 under those digits to get:

```
  23
+ 51
  74
```

The carry comes in when we add two numbers and get a result too large to fit into a single digit, as in 6 + 8. In this case we have "four, carry the one," thus:

```
  1
   6
+  8
  14
```

Notice that the carried one drops down into the next place in the number. If we were adding 16 and 8, the carry would be added to the 1 in 16, resulting in an answer of 24.

Now all of this is very elementary, but it demonstrates all the necessary actions to add in binary. Here is the "truth table" for addition in binary:

```
0+0= 0
0+1= 1
1+1=10
```

The first three additions are "common sense," but the final one, 1 + 1 = 10 deserves a second look. We know that one plus one equals two, but we're working in binary, so two is expressed as one-zero, or 10. This is also equivalent to "zero, carry the one," since "10" cannot fit in a single digit.

Let's run through a sample addition in binnary:

```
     1111
   00000101      ( 5)
 + 00001011      (11)
   00010000      (16)
  (87654321)
```

1. 1+1=0, carry the one
2. 0+1=1, plus carry of 1 gives 0, carry the one
3. 1+0=1, plus carry of 1 gives 0, carry the one
4. 0+1=1, plus carry of 1 gives 0, carry the one
5. 0+0=0, plus carry of 1 gives 1 — no carry!

As always, since we are working with eight-bit bytes, we fill all unused digits with zeros. This is important.

As you can see, a single one can cause a whole string of carries, almost like a chain reaction. It is possible that the carry could be continued past the seventh bit (marked 8 above). Therefore, most microprocessors have a special register, called the *carry bit* to hold and signal this runaway bit. This bit is essential in adding multibyte numbers, which we will cover in Part Three. Let's try another addition.

```
    11
   00011101      (29)
 + 00110010      (50)
   01001111      (79)
  (87654321)
```

This time we have an interesting effect of the carry. In step 5, we get 1 + 1 = 0, carry the one. In step 6, we add 1 + 1 + the carry of 1 to get 1, carry the one (1 + 1 + 1 = 11). The carry comes to rest at step 7. Incidentally, I have numbered the bits from 8 to 1 for convenience. In reality, they are numbered from 7 to 0, the exponents of the powers of two. (Bit 6 = 2^6 = 64).

You now have the necessary information to add in binary, but in order for it to really "sink in," you will have to practice it until it becomes clear. You can make up your own exercises by randomly stringing a series of ones and zeros together to form two eight-digit numbers. Then add them in binary. To check your answer, convert the addends and the answer into decimal, which you can easily verify.

When you are confident that you can add in binary, you are ready to grasp this section on subtraction. When we perform subtraction in our normal, base ten system, we are really just adding the two numbers. For example, 8 - 5 is equivalent to 8 + (-5). -5 is pronounced "negative five." It is assumed that you are aware of negative numbers, as it is taught as early as sixth grade, but we all can forget, right? All that is necessary is to know that, when you add a negative and a positive number, you can get the same result as subtracting the smaller number from the larger number, and giving the answer the sign of the larger number. When you add two negative numbers, the answer is the same as adding the numbers, disregarding their sign (the absolute value), and then giving the sum a negative sign (e.g. (-4)+(-3)=-7). Yet believe it or not, subtraction in binary is even easier than in decimal, as a comparison will show.

First we have to know how a negative number is expressed in binary. Since a binary number is composed of ones and zeros, there is no place for the minus sign. Therefore, the highest bit, bit seven, is used to show that the number is negative. Most microcomputers use a technique called "two's complement" to convert a number into its negative equivalent. If you add numbers using two's complement, the subtraction will be performed automatically. Two's complement has two steps — forming the complement, and adding 1 to it. Numbers properly represented using two's complement are called *signed binary*.

Let's form the signed binary equivalent of -5. Here is the binary equivalent of five: 00000101. To complement it, we turn all the zeros into ones, and all the ones into zeros to get: 11111010. Next we add 1 to it to get:

```
   11111010
 + 00000001
   11111011
```

Positive numbers in signed binary are expressed normally, with the restriction that they must not be greater than 127. If they were, bit seven would be "on," and the number would look as if it were negative. The number 205 in straight binary is 11001101. This is also -51 in signed binary. You can find the value of any negative number in signed binary by running it through the two's complement routine again. You'll get the positive value of the number. Similarly, if you try to make any number larger than 128 negative, it will end up positive. Therefore, in signed binary, the value in the byte must be between -128 and positive 127. Now that we have our background, let's try out our skills.

```
    Subtract: 43-11.         43=001010011

-11=  00001011
      11110100     complement
    + 00000001     plus one
      11110101

    Add 43 and -11:

      1111111
      00101011       43
    + 11110101      -11
      00100000      -32          C:1
```

Notice that the carry was swept out of the byte (C:1). C: represents the imaginary carry register. This carry should be always disregarded in two's complement subtraction. The most wonderful thing about subtraction in binary is that it is seemingly "automatic." But once again, for complete understanding, you must practice subtraction until you feel sure of your comprehension. For this purpose, exercises are once more included at the end of this article.

Next time, we'll learn about *multibyte* numbers and even get into a wee bit of MACHINE LANGUAGE!

Answers to exercises in PART ONE:

1. a) 21 b) 51
c) 60 d) 255

2. a) 00110100 b) 11101010
c) 01000010 d) 00001111

3. The complete chart to sixteen bits:

```
32768   16384   8192   4096   2048   1024
512   256   128   64   32   16   8   4   2   1
```

EXERCISES

1. Add:

```
a)   00101011        b)   01000011
   + 00000111           + 00011000

c)   00111000        d)   10011010
   + 10100111           + 00111001
```

2. Convert to binary and add:
a) 20+11
b) 18+56
c) 29+47
d) 32+64

3. Complement only:
a) 01010110 b) 01100011

4. Form the two's complement
a) 01111001 b) 10111111

5. Convert into signed binary:
a) -14 d) 108
b) 22 e) -9
c) -134

6. Convert to binary and subtract:
a) 56-18 b) 99-33
c) 58-78 c) -105 -12

7. Why is -56 equal to 200? (Trick question)

©

TOLL FREE
Subscription
Order Line
800-345-8112
In PA 800-662-2444

THE apple® GAZETTE

Animating Applesoft Graphics

Leslie M. Grimm
Mt. View, CA

Animating graphics can add a special plus to your BASIC program. A previous article (**COMPUTE!** #14) described how to animate low-resolution graphics in Integer BASIC. A method is described here to do animation of either high-resolution or low-resolution graphics in Applesoft BASIC.

Before beginning, however, a few words comparing the two BASICs for this purpose are in order. Integer BASIC is much faster than Applesoft. This is because the Applesoft interpreter must perform time-consuming manipulations of floating point arithmetic, whereas the Integer BASIC interpreter ignores everything to the right of a decimal point. The effect of all this is that Applesoft graphics routines run about half as fast as Integer routines. This can be crucial in animation.

In general, if the object to be animated is very large (bigger than ¼ of the low-resolution screen area or bigger than about 20 x 20 dots in high-resolution) you will get better results in Integer. However, choice of Applesoft may be a matter of necessity for a variety of reasons. By keeping animated objects small and simple, and observing other speed-increasing tips mentioned below, you can get very nice effects.

Designing The Figure

For the low-resolution example listing below, the figure of a flying bird was chosen. The high-resolution example uses a simple shape (square) for the sake of brevity in this article, but you could modify the bird or make any shape you desire for high-res.

Whatever shape you choose, your first step is to draw the figure in various states of motion. Use graph paper, and number the squares as shown in Figure 1. (This applies to low- or high-res shapes.). Note that, for the flying bird, three different positions simulate the action of flying.

Because the figure will be moving about on the screen, you need to use relocatable coordinates in your plotting routine. Consider the square in the upper-left-hand corner as X = 0, Y = 0. Then specify all other points relative to that point. For example, a point five squares to the right and three squares down would be called X + 5,Y + 3.

You should also think about the most economical way to draw the figure. In the case of the bird, you can see that the body is the same for all three drawings. One subroutine was made for it, and another for the wing in its upward position, and still another for the wing in its downward position. To draw the bird with its wing up, the program does a GOSUB to the body routine (at 100) followed by setting hue to 2 (blue) and issuing a GOSUB to the wing-up routine (at 110). Note that the subroutines for wing up and wing down use a variable (hue) for color. This way the same subroutine can be used to draw (hue = 2) or erase (hue = 0) the wing.

In writing the code it is important to keep speed of execution in mind. As much as possible you should put many statements on a single line, separated by colons. Use HLIN and VLIN commands instead of a lot of HPLOTs. Locate your graphics subroutines at low line numbers.

Smooth Animation

The basic technique in animation is to draw the figure at a certain location on the screen, then erase it and redraw it at a new location. (An alternative method is to draw the figure at location one, redraw it at location two, and erase the parts that are left over from location one. If you know that your figure will always move exactly the same number of spaces each time it is redrawn the latter method is preferable. It could work reasonably well without page flipping also, but, because it is not the most general case, it is not demonstrated here.)

For the flying bird, the erase procedure was done with two routines. Line 150 draws the body in color = 0 (black), and then hue is set to zero and the appropriate wing routine is used. Note that if you wanted to use a colored background the erase routine could use the color of the background rather than zero.

If that were all you did, though, you would probably be disappointed in the results. This is because you would be watching the figure being drawn and erased on the screen. This is distracting and can be avoided by the technique of "flipping pages." Pages can be flipped for either low-resolution or high-resolution graphics, and the methods to do this are described separately below.

The technique for flipping pages is similar for low- and hi-resolution graphics. There are two graphics screen pages for low-resolution graphics (beginning at $400 and $800 respectively) and two screen pages for high-resolution graphics (beginning at $2000 and $4000 respectively). Your program will display one page to the user while erasing and drawing "behind the scenes" on the other page.

In low-resolution graphics it is not possible to draw directly to the second screen page. Drawings can only be placed on screen page two by first making them on page one and then calling a routine in the Apple monitor to move the contents of page one onto page two.

You will need a short assembly language routine to do the move for you. The subroutine beginning at line 10000 pokes this assembly language routine in memory. All you need to do is CALL the routine when you need it.

(A description of how the routine works follows, but you don't need to know how it works to use it. Just skip on to the next paragraph if you wish.) The LDA $C054 at line $C00 causes the Apple to display page one. The lines from $C03 to $C15 specify that the contents of memory locations $400 through $7FF (graphics page one) are to be moved to the region from $800 to $BFF (graphics page two.) Line $C17 sets a counter (Y register) to zero, and the next line does a Jump to SUBRoutine (JSR) at $FE2C — the move routine in the Apple monitor. The move routine transfers the contents of page one to page two very quickly. Line $C1C causes page two to be displayed, and the last line ReTurnS you to your BASIC program.

Bird In Flight

Line 10 sets text mode (in case a previous program had left the machine set to graphics mode) and clears the screen Line 20 POKEs the assembly language routine in via the subroutine beginning at line 10000. Line 40 branches around the graphics subroutines to the start of the animation program. (The graphics subroutines were intentionally placed at low line numbers for speed of execution.)

The animating program first clears the screen (page one), sets initial values for X and Y, and calls the move routine (CALL 3072). The user will now be looking at page two, which is blank. Next, line 1010 draws the figure in its initial position (wing down) behind the scenes on page one. It then calls the move routine. Remember that the move routine displays page one while it is copying page one onto page two, and then flips to page two. The user only sees the finished drawing, first on page one and then on page two. The flip between pages doesn't show.

While that drawing is being displayed the original figure on page one is erased (line 1020). The value for X is changed and the figure is redrawn in a new position (wing up) and a new location (line 1030). Once again the move routine is called to put the new drawing on page two and show it to the viewer.

Line 1040 erases the wing-down bird, moves the bird over and up, and draws just the body. Then it performs the move and flip. In line 1050 the body is erased, and the bird is drawn with wing down in its next location. The move and flip is called again. This process is repeated several times in a FOR ... NEXT loop.

The last lines of the routine restore the display to graphics page one. The cursor is VTABbed to line 21 so that it will be visible when the program ends. The POKEs instruct the computer to locate the next Applesoft program at the normal location ($800). (See below)

In entering and debugging a program that flips pages you may occasionally get "stuck" on page two due to a programming error. When this happens you will hear the beep that accompanies an error message, but no message will show and there will be no cursor. Just type "POKE 16300,0 to restore the display to page one and see your error message.

Relocating

There is one more step required before you can actually run this program. Page two of low-res graphics occupies the same place in memory that your Applesoft program normally occupies. Your only alternative is to relocate your Applesoft program. To do this, before you load your program you must change the values of the "program start" pointers to a new value. This will cause your program to be loaded in at a different place than usual.

The Applesoft program could be relocated to many possible places in memory. In this example it was located at the end of the assembly language subroutine. The assembly language subroutine was placed just above the second low-res graphics page. Alternatively, one could put the assembly language routine at $300 (decimal 768), but since this area is often needed for music routines, it was left free here.

There are several ways to relocate the program. One way is to type the following commands before running your program:

POKE 103,33
POKE 104,12
POKE 3104,0

The first two POKEs place the starting address of the program in memory. The third POKE sets the first byte of the program location to zero, which must be done in order for the Apple to find the program's beginning.

Alternatively, you can write a short program to do the POKEs for you. A sample listing is Program 1.

(A third method, which incorporates the relocating program as a subroutine of the main program, will not be explained here for the sake of brevity.)

Whichever method is used to relocate the program, it is a good idea to restore the pointers to their usual values at the end of your program. The next Applesoft program will then load into the normal area of memory. This is shown at the end of the example program.

Flipping between high-resolution pages is easier than flipping in low-resolution graphics because it is possible to draw directly on either page. Also, it is not necessary to relocate your Applesoft program. However, only very small drawings can be animated in BASIC, due to speed limitations. Program 3 moves a very small, simple shape (square) diagonally across the screen, flipping pages between each move.

Line 10 clears both hi-res pages and sets the screen to full-screen graphics. Full screen is necessary to prevent text "garbage" from appearing at the bottom of screen page two.

The subroutine at line 100 draws or erases the square, depending on the value given to hue. A value of 5 sets the color to orange, and 0 is black. Line 1000 sets up the original values for X and Y, and causes page two to be displayed (POKE–16299,0).

The value POKEd into location 230 determines whether your program draws on hi-res graphics page one or two. To draw on page one this value must be 32 ($20). To draw on page two, set it to 64 ($40). Note that you could also simply specify HGR for page one or HGR2 for page two, but these commands include an implicit "clear screen" which would erase the whole screen and take far too long.

As in the low-res animation process, the program displays only finished drawings to the viewer while it erases and redraws figures on the undisplayed pages. Line 1002 directs the drawing process

ECHO SERIES™ *SPEECH SYNTHESIZERS*

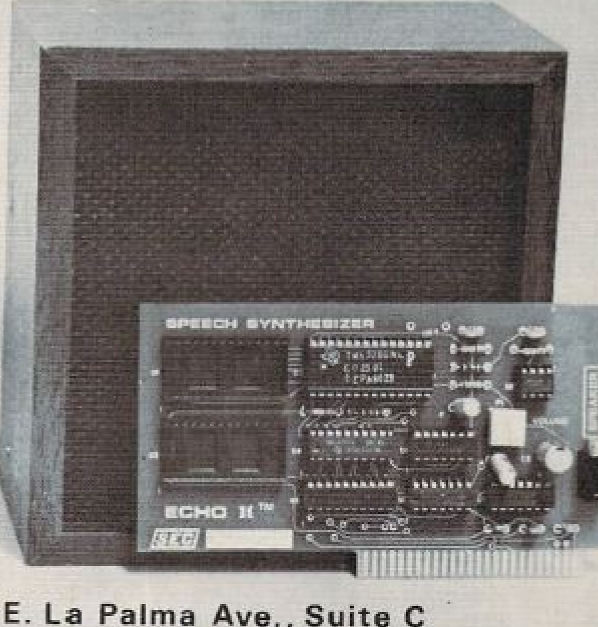
COMPUTERS ARE SPEAKING OUT!

Now you can add intelligible speech to your computer without using vast amounts of memory! The ECHO][™ speech synthesizer for the Apple* is the first of a series of synthesizers based on the same technology that made the Speak & Spell** a success.

The initial operating system allows the creation of your own vocabulary with phonemes (word sounds) while using very little RAM memory (approx. 800 bytes + 20 bytes/word). Enhanced operating systems and vocabulary ROMs will be offered as they become available.

The ECHO][™ comes complete with speaker, instruction manual, and a disk containing a speech editor, sample programs, and a sample vocabulary. Suggested list price is $225.

See your dealer or contact:



SEC STREET ELECTRONICS CORPORATION

3152 E. La Palma Ave., Suite C
Anaheim, CA 92806 (714) 632-9950

*Trademark of Apple Computer **Trademark of Texas Instruments to page one, but it will not be seen since page two is being displayed.

Again, as in the low-res animation routine, a FOR ... NEXT loop is used. Line 1010 sets the color to orange and the GOSUB 100 draws it on page one. The POKE–16300,0 flips the display to page one when the drawing is finished. To the viewer, the drawing seems to pop onto the screen.

Line 1020 first resets X and Y to the previous location so that the last square on page two can be erased. Location 230 is set to 64 so that drawing will be done on page two. X and Y are then advanced to the new location, color is set to orange again, and the new square is drawn. Finally, the display is flipped back to page two. The viewer sees the square slide to a new location.

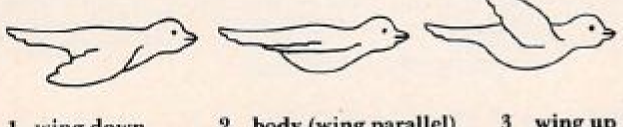
Line 1030 sets drawing to occur on page one again, erases the square there, and sets X and Y to the location for the next square. When the NEXT J instruction in Line 1040 is encountered, the program will jump back to line 1010, which will actually draw the square.

Line 2000 restores the display to page one, and ends. One could add the command TEXT before END to restore the viewer to text mode.

This method for high-resolution animation is not as satisfactory as an assembly language routine would be, but could be useful in many simple applications. Another possibility for a simpler way to use this method would be to have two pictures, (one on each page) showing different positions of the same figure. For example, one could have a Jumping Jack with arms up in one and arms down in the other picture. These could be large, elaborate drawings. By flipping between the two pages (POKE –16299,0, then POKE –16300,0) many times the Jumping Jack would appear to swing its arms up and down. In practice, it would probably be necessary to have a short delay between successive flips for this application.

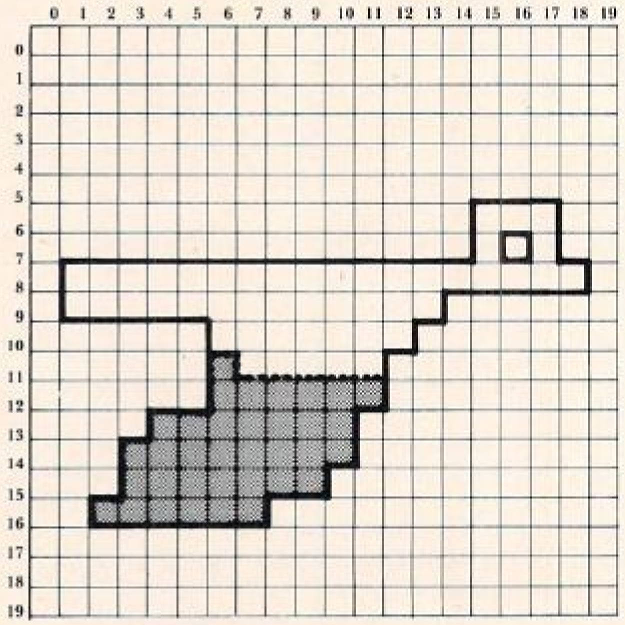
Many other techniques of animation can be employed, but these methods should provide a starting point for the beginning or intermediate level Applesoft BASIC programmer.

Figure 1. Sketch of flying bird.



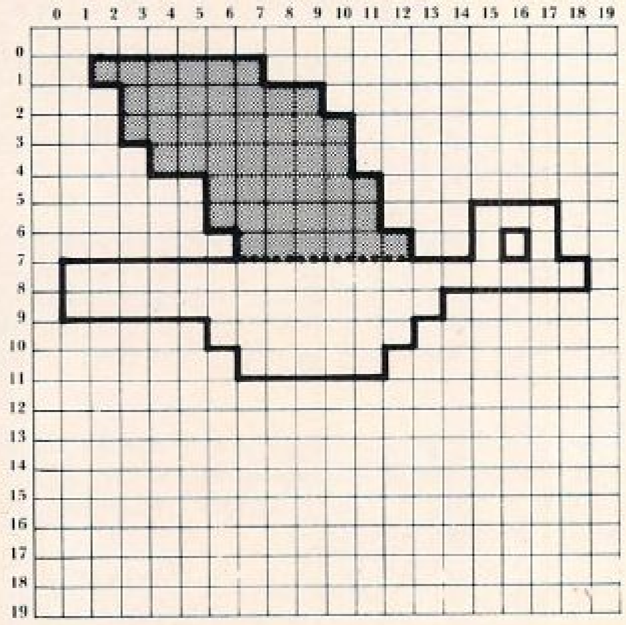
1 wing down 2 body (wing parallel) 3 wing up

Figure 2. Wing down and body.



.... = bottom of body

Figure 3. Wing up and body.



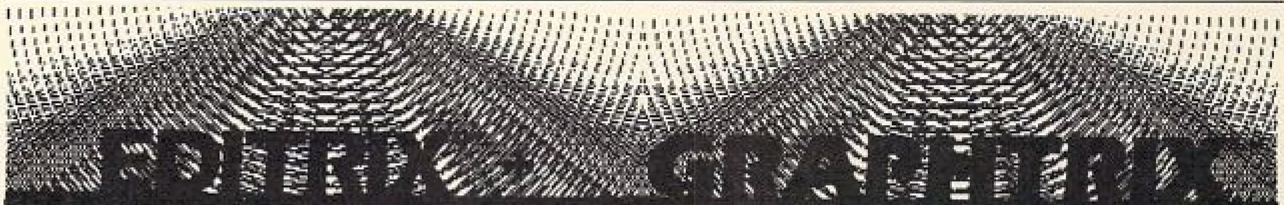
... = top of body

Program 1. (50)

```
5  REM  BIRD LOADER PROGRAM
10  TEXT : HOME : VTAB 10
20  FLASH : HTAB 17: PRINT "LOADING": NORMAL
30  PRINT : PRINT : HTAB 13: PRINT "BIRD IN FLIGHT"
40  POKE 103,33: POKE 104,12: POKE 3104,0: REM  RELOCATES NEXT APPLESOFT
     PROGRAM TO LOAD AT $C20
50 D$ =  CHR$ (4): PRINT D$;"RUN BIRD IN FLIGHT"
```

Program 2. (10010)

```
3  REM  BIRD IN FLIGHT
5  REM   POKE 103,33, POKE 104,12, POKE 3104,0 TO RELOCATE PROGRAM BEFORE
      RUNNING
10  TEXT : HOME
20  GOSUB 10000: REM  POKE IN MOVE AND FLIP ROUTINE
40  GOTO 1000
99  REM  ** GRAPHICS SUBROUTINES **
100  COLOR= 2: HLIN X + 1,X + 18 AT Y + 8: HLIN X + 1,X + 13 AT Y + 9: HLIN
     X + 6,X + 12 AT Y + 10: HLIN X + 7,X + 11 AT Y + 11
102  HLIN X + 15,X + 17 AT Y + 6: HLIN X + 15,X + 17 AT Y + 7: COLOR= 0: PLOT
     X + 16,Y + 7: COLOR= 1: HLIN X + 17,X + 18 AT Y + 8: RETURN : REM BO
     DY
110  COLOR= HUE: HLIN X + 2,X + 7 AT Y + 1: HLIN X + 3,X + 9 AT Y + 2: HLIN
     X + 3,X + 10 AT Y + 3: HLIN X + 4,X + 10 AT Y + 4
112  HLIN X + 6,X + 11 AT Y + 5: HLIN X + 6,X + 11 AT Y + 6: HLIN X + 7,X
      + 12 AT Y + 7: RETURN : REM  WING UP
```



EDITRIX + GRAPHTRIX = THE MOST POWERFUL WORD PROCESSOR
THIS SIDE OF A NEWSPAPER COMPOSITION ROOM

EDITRIX™
TEXT EDITOR

EASY TO USE
- Friendly, COMPLETE instructions that you or your your secretary can understand.
- Easy to remember 1 or 2 keystroke commands.
- Fully Menu Driven.

POWERFUL
- Full Cursor Control.
- Full Print Size and Emphasis Control.
- Underline - Superscript - Footnotes.
- Search - Replace - Block Move.
- Automatic Graphic Insertion and Formatting.
- All Justify Modes and Full Margin Control

FLEXIBLE
- 40 or 80 Column Display.
- Printout through GRAPHTRIX to 11 different Printers WITHOUT CHANGING YOUR TEXT FILE!

REQUIRES: Apple II with 48K, Applesoft in ROM, DOS 3.3 and the GRAPHTRIX Matrics Graphics System

GRAPHTRIX™
TEXT PRINTER AND GRAPHICS SCREEN DUMP

EASY TO USE
- Complete READABLE documentation.
- Fully Menu Driven.
- Self-running Introduction and Demonstration.

POWERFUL
- Graphic Magnification, Normal/Inverse, Page Centering, Hi and Low Crop Marks, Title String.
- Automatic Formatting of Graphics in your Document.
- Print Size, Emphasis, Underline, Superscript, Footnotes, Chapters, controlled from your text file.

FLEXIBLE
- Prints ANY HI-RES Graphic your Apple II can create.
- Formats Text files from Applewriter OR EDITRIX.
- Use as a Menu Driven Screen Dump OR from in YOUR OWN Applesoft Program.
- Compatible with 11 different Matrix Line Printers AND 7 different Parallel Interface Cards.

REQUIRES: Apple II with 48K, Applesoft in ROM, DOS 3.3 and one of the following line printers: EPSON MX-70/MX-80/MX-100, ANADEX 9500/9501, IDS 440G/445G/460G/560G, CENTRONICS 739, MPI 88G, SILENTYPE.

FROM DATA TRANSFORMS, INC., THE GRAPHICS LEADER

EDITRIX and GRAPHTRIX are the trademarks of Data Transforms Inc., a division of Solarstatics Inc.
Apple II and Applewriter are trademarks of Apple Computer Inc.
(c) Copyright 1981 Data Transforms, Inc. 906 E. Fifth Ave. Denver, CO 80218 (303) 722-8774
All Rights Reserved

```
120  COLOR= HUE: PLOT X + 6,Y + 11: HLIN X + 6,X + 11 AT Y + 12: HLIN X +
     4,X + 10 AT Y + 13: HLIN X + 3,X + 10 AT Y + 14
122  HLIN X + 3,X + 9 AT Y + 15: HLIN X + 2,X + 7 AT Y + 16: RETURN : REM
     WING DOWN
150  COLOR= 0: HLIN X + 1,X + 18 AT Y + 8: HLIN X + 1,X + 13 AT Y + 9: HLIN
     X + 6,X + 12 AT Y + 10: HLIN X + 7,X + 11 AT Y + 11: HLIN X + 15,X +
     17 AT Y + 6: HLIN X + 15,X + 17 AT Y + 7: HLIN X + 17,X + 18 AT Y +
     8: RETURN : REM ERASE BODY
999  REM  ** ANIMATION DRIVER **
1000  GR : HOME :X = 0:Y = 20: CALL 3072
1010  GOSUB 100:HUE = 2: GOSUB 120: CALL 3072
1020  FOR FLY = 1 TO 4: GOSUB 150:HUE = 0: GOSUB 120:X = X + 2
1030  GOSUB 100:HUE = 2: GOSUB 110: CALL 3072
1040  GOSUB 150:HUE = 0: GOSUB 110:X = X + 1:Y = Y - 1: GOSUB 100: CALL 3
     072
1050  GOSUB 150:X = X + 2:Y = Y - 1: GOSUB 100:HUE = 2: GOSUB 120: CALL 3
     072
1060  NEXT FLY: POKE  - 16300,0
1200  VTAB 21: POKE 103,1: POKE 104,8: POKE 2048,0: END : REM  RESET PROG
     RAM START POINTERS TO NORMAL VALUE
9990  REM  ** ASSEMBLY LANGUAGE ROUTINE
9992  REM  COPIES LO-RES GRAPHICS PAGE ONE
9994  REM  TO PAGE TWO WITH PAGE FLIPPING
9996  REM  LOCATED AT $C00 (3072)
10000  FOR I = 3072 TO 3103: READ CODE: POKE I,CODE: NEXT I: RETURN
10010  DATA    173,84,192,160,0,132,60,169,4,133,61,169,255,133,62,169,7,
     133,63,169,8,133,67,132,66,32,44,254,173,85,192,96
```

Program 3. (2000)

```
5  REM  ANIMATED SQUARE
6  REM  HI-RES ANIMATION DEMO
10  HOME : HGR2 : HGR : POKE  - 16302,0: REM   FULL SCREEN
20  GOTO 1000
99  REM  ** DRAW SQUARE **
100  HCOLOR= HUE: FOR I = Y TO Y + 10: HPLOT X,I TO X + 10,I: NEXT I: RETURN
999  REM  ** ANIMATION DRIVER **
1000 X = 50:Y = 50: POKE  - 16299,0: REM  DISPLAY PAGE TWO
1002  POKE 230,32: REM   DRAW ON PAGE ONE
1005  FOR J = 1 TO 20
1010 HUE = 5: GOSUB 100: POKE  - 16300,0: REM  DISPLAY PAGE ONE
1020 X = X - 2:Y = Y - 2:HUE = 0: POKE 230,64: GOSUB 100:X = X + 4:Y = Y +
     4::HUE = 5: GOSUB 100: POKE  - 16299,0
1030  POKE 230,32:X = X - 2:Y = Y - 2:HUE = 0: GOSUB 100:X = X + 4:Y = Y + 4
1040  NEXT J
1050  POKE  - 16301,0: REM  RESTORE MIXED TEXT AND GRAPHICS MODE
2000  POKE  - 16300,0: VTAB 22: END
```

Program 4. (0C1F) Assembly Language Routine Flip And Move

```
0C00-   AD 54 C0    LDA   $C054         0C11-   85 3F      STA   $3F
0C03-   A0 00       LDY   #$00          0C13-   A9 08      LDA   #$08
0C05-   84 3C       STY   $3C           0C15-   85 43      STA   $43
0C07-   A9 04       LDA   #$04          0C17-   84 42      STY   $42
0C09-   85 3D       STA   $3D           0C19-   20 2C FE   JSR   $FE2C
0C0B-   A9 FF       LDA   #$FF          0C1C-   AD 55 C0   LDA   $C055
0C0D-   85 3E       STA   $3E           0C1F-   60         RTS
0C0F-   A9 07       LDA   #$07
```

©

Apple Authors:

COMPUTE! is looking for good applications articles on Apple.

Paddle-Adapple

I/O ADAPTOR FOR YOUR APPLE

- COMPATIBLE WITH ALL JOYSTICKS AND PADDLES
- COMPATIBLE WITH ALL SOFTWARE
- GIVES YOU **FOUR** PUSH-BUTTON INPUTS
- UNIQUE "JUMPERS" SOCKET ALLOWS YOU TO CONFIGURE THE **PADDLE-ADAPPLE** TO MEET YOUR SPECIFIC NEEDS
- SELECT ONE OF TWO DEVICES PLUGGED IN or
- USE FOUR PADDLES SIMULTANEOUSLY
- **BPI**™ USERS CAN HAVE **BPI** DEVICE AND PADDLES PLUGGED IN SIMULTANEOUSLY
- EXCHANGE X AND Y AXES IN JOYSTICKS
- ALL STROBES, ANNUNCIATORS, AND POWER LINES AVAILABLE ON BOTH CONNECTORS
- SUPPLIED WITH CABLE (ONE FOOT)
- ADHESIVE FOAM TAPE SUPPLIED
- SUPPORTS SHIFT KEY MODIFICATION

APPLE is a trademark of Apple Computer, Inc
BPI is a trademark of BPI Systems, Inc.
PADDLE-ADAPPLE is a trademark of Southern California Research Group

AVAILABLE FROM YOUR COMPUTER DEALER
or
SOUTHERN CALIFORNIA RESEARCH GROUP
POST OFFICE BOX 2231-C
GOLETA, CA 93118
(805) 685-1931

$29.95

Add $1.50 for shipping and handling
CA residents add tax.

MASTERCARD AND VISA ACCEPTED
DEALER INQUIRIES INVITED

utility city

21 of our most-asked-for Apple Utilities
by Bert Kersey 48K Applesoft ROM

For YOUR Big Apple—Our 21 most-asked-for Apple Utilities on one big disk—**List Formatter** makes properly spaced & indented listings with printer page breaks. Each program statement is on a new line with If-Then's & Loops called out; a great de-bugger! **Catalog in any number of columns** & any page-width to CRT or printer. Automatically post the Run-Number & last-used Date in your programs. **Make any command invisible** in your listings; Access program lines in memory for garbage repair & "illegal" alteration; Quickly sort & store info on disk; Run any Applesoft file while another stays intact; Move lines within your programs; Renumber to 65535; Save inverse, INVISIBLE & trick file names; Convert dec to hex & binary, or INT to FP; Append programs; dump the text screen to ANY printer . . . More too: **21 Programs Total!**

PLUS . . . APPLE TIP BOOK NUMBER THREE!
40 PAGES of new tips, tricks & articles—"DOS Trickery," "Copy Stoppers," "Programming the Reset Key" & much more. Plus understandable explanations of how each U-City Program works—One of the best Apple learning tools on the market today!

$29.50 +$1.50 shipping
✓ **Utility City** on Applesoft Disk
✓ 40-page Beagle Bros Apple Tip Book #3
✓ Apple PEEKS, POKES & POINTERS Chart

alpha plot

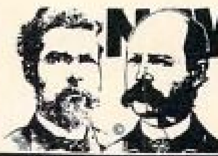
Hi-Res Apple Graphics/Text Utility
by Bert Kersey & Jack Cassidy 48K REQUIRED

HI-RES DRAWING: Create hi-res pictures & charts, **appendable to your programs.** Keyboard or Paddle control; Optional Xdraw Cursor (see lines before you draw!); Any color mix or REVERSE (opposite of background); Circles, Boxes & Ellipses, filled or not. Bonus Programs too—**SCRUNCHER** stores hi-res in as little as 1/3 normal disk space. **SHIFTER** transfers any portion of the hi-res screen. Also superimpose hi-res images and convert Hi-Res to Lo-Res & back for fascinating abstracts!

HI-RES TEXT: Beautiful upper & lower case with Descenders; color or reverse; Positionable anywhere (NOT restricted by Htab & Vtab). Professional looking **PROPORTIONAL SPACING!** Adjustable Type Size, Leading (line spacing) & Kerning (letter spacing). Multi-directional typing; up, down, even backwards!

PLUS . . . APPLE TIP BOOK NUMBER FOUR!
40 PAGES of tips for making the most of your Apple's advanced graphics capabilities. Alpha instructions plus shape table tips, new titling & color tricks and fascinating animation experiments!

$39.50 +$1.50 shipping
✓ **Alpha Plot** on Applesoft Disk
✓ 40-page Beagle Bros Apple Tip Book #4
✓ Apple PEEKS, POKES & POINTERS Chart



NEW Apple® Utility Disks!

With Free PEEKS & POKES Wall Chart!

Beagle Bros MICRO SOFTWARE
4315 Sierra Vista
San Diego,
CA 92103
714-296-6400

WITH EACH ORDER, YOU WILL RECEIVE . . .
☐ **Dos Boss, Utility City** or **Alpha Plot** on Applesoft Disk
☐ A 40-page **Apple Tip Book** (each entirely different)
☐ An 11x17 Apple **PEEKS, POKES & POINTERS** Chart

BONUS WITH EACH DISK!
POKE your Apple all night long with this handy 11 × 17 reference poster! The most useable PEEKS, POKES, POINTERS & CALLS scrounged up from every source imaginable! A great companion to our original Apple Command Chart.

dos boss

DISK COMMAND EDITOR
by Bert Kersey & Jack Cassidy

A classic utility you will ENJOY! **Rename Commands:** "Catalog" can be "Cat," etc. **Save-Protect your programs;** Unauthorized copy attempts produce "Not Copyable" message. 1-Key Program Selection from catalog (extremely handy!). Catalog Customizer—Change Disk Volume message to your title, Omit or alter file codes, Catalog by file type, etc. Rewrite Error Messages—"Syntax Error" can be "Oops!" or anything you want!

All of Dos Boss's change feaures may be appended to your programs, so that anyone using your disks on any Apple (booted or not) will be formatting DOS the way you designed it!

PLUS . . . APPLE TIP BOOK NUMBER TWO!
With valuable DOS information. "Poking Around in DOS," "Making Programs Unlistable," "Two-Sided Disk Tips," "Care of Your Drives," "Creative Peeking & Poking," "3.2 vs 3.3," etc.

ONLY $24. +$1.50 Shipping
✓ **Dos Boss** on Applesoft Disk
✓ 40-page Beagle Bros Apple Tip Book #2
✓ Apple PEEKS, POKES & POINTERS Chart
MINIMUM 32K REQUIRED

24 hour TOLL FREE Order Desk:
MasterCard, VISA or COD orders, call:

Nationwide:	800-854-2003	ext. 827
California:	800-522-1500	ext. 827
Alaska/Hawaii:	800-854-2622	ext. 827

(Orders only please. Questions, phone 714-296-6400.) COD orders, add $3.)

AT YOUR APPLE DEALER NOW OR ORDER BY PHONE AT ANY HOUR... TOLL FREE!

Ask around about Beagle Bros Software.
Our programs are practical, friendly & user-oriented, and always come with an amazing amount of documentation & juicy Apple information. If your dealer doesn't have our stuff, get on his case about it, or order directly from us. We ship all disks & books WITHIN 48 HOURS.

Or order by mail (use this coupon or separate sheet)

RUSH! The disks checked below
Plus The Tip Book and PEEKS & POKES Chart!

☐ **UTILITY CITY** ($31) ☐ **ALPHA PLOT** ($41) ☐ **DOS BOSS** ($25.50)

(Prices include First Class Shipping; Calif. residents add 6%.)

Specify:
NAME$= ______________ ☐ **3.2** ☐ **3.3**
ADDR$= ______________
CITY$= ______________ **ZIP=** ______

VISA

VISA/MASTERCARD: Include Account No. & Expiration Date.

master charge

Mail U.S. check, money order or Visa/MC Numbers
to: **BEAGLE BROS, Dept. 5**
4315 Sierra Vista,
San Diego, CA 92103
Telephone (714) 296-6400.

Please add $4.50 for shipping outside US or Canada.
COD orders, add $3. California residents, add 6%.

Programming The RESET Key The Easy Way

Richard Cornelius, Wichita, KS

On the Apple Computer the RESET key, to most users, is a magical key that provides an instant means to get out of any program. Usually a person begins to modify the RESET function only after learning machine language. Here is a method of making the RESET key do anything (well, almost anything) that you want it to do on an Apple II Plus, and you don't need *any* knowledge of machine language.

First let's examine what the RESET key does. When the RESET key is pressed the currently running program is interrupted; the screen display is set to text page 1; output to the screen is set to NORMAL; the text window is set to the complete screen; the cursor is moved to the bottom of the page; a beep sounds; accessory I/O is shut down; and then the computer looks in locations 1010 and 1011 in memory to see where it should go next for instructions. When the computer is turned on, the contents of these two locations are automatically set such that when RESET is pressed the computer is returned to immediate mode in BASIC. Changing these locations to make the computer go to different places for instructions involves only POKEs to positions 1010 and 1011 and a CALL-1169.

Where should the computer be sent? Starting at position 768 there is some room that is reserved for short machine language programs, and that is where we shall send it. (Don't worry — you don't need to know any machine language.) POKEs to seven bytes are used to make the RESET key run a BASIC program starting at the *second* line of code. When the first line of the program makes the program jump around the second line, then the second line will *only* be executed when RESET is pressed.

The program will help you understand how the RESET key can be used to execute any BASIC statements that can be put into a program. When the program is RUN, statements 110 through 190 are jumped over so that lines 200 through 260 are the first statements in the program that perform any tasks. These lines fix the RESET key so that the computer will go to line 110 when the RESET key is pressed. The length of the very first statement is critical. As long as it has a three-digit number after the GOTO, the RESET key will operate as desired. Changes in the length of the statement will likely mean that the RESET key will send the computer to some nonsense location. Placing a REM statement (or any other statement) before line 100 will have the same effect. Modifying the DATA statement in line 230 to accommodate changes in the length of that first statement is not difficult, but, unless you understand what to do, you had better not make any changes.

Lines 270 through 310 constitute a dummy BASIC program to show that the program is being RUN. Statements 110 through 190 tell the computer what to do when the RESET key is pressed. Lines 140 through 190 can be changed to make the RESET key do whatever you want it to do. In this example, the program is simply rerun from the beginning, but you can make lines 140-190 do whatever you wish. Lines 120 and 130 should not be changed since they fix up some things that are undone by the short machine language program that is POKEd in, but omit line 130 if you don't have a disk drive. If you should want to "turn off" the changes to the RESET key so that it behaves normally, simply POKE 1010,3: POKE 1011,244: POKE 1012,69 if you have no disk drive or POKE 1010,191: POKE 1011,157: POKE 1012,56 if you do have a disk drive.

For those who don't wish to stray from BASIC, this short program contains all that is needed to make the RESET key do almost anything. Take an existing program and add it starting at line 280 to the program. In lines 140-190, put statements that you wish to be executed when the RESET key is pressed. You can thus program the RESET key in BASIC without knowing any machine language. For those who are interested in straying just a little from BASIC, the final paragraphs explain the details of what is happening.

Positions 1010 and 1011 (hex 3F2 and 3F3) contain the low and high bytes of the location that the RESET key makes the computer jump to after it performs a fixed set of operations. The POKEs in statement 210 change this location from 40383 (hex 9DBF) to 768 (hex 300). Before the computer performs this jump, it looks at the "power-up" byte, position 1012 (hex 3F4), to see whether the value at this location equals an exclusive OR of the value in position 1011 (hex 3F3) with the constant 165. If the values correspond properly, the computer believes that it has *not* just been turned on and it executes a jump to the specified location. If the values do not properly correspond, the computer thinks that it has just been turned on, and it will attempt to reboot the disk if a disk controller card is present. The CALL-1169 in statement 210 properly sets this power-up byte.

The DATA statement in line 230, coupled with the POKE statement in the FOR...NEXT loop in lines 240 through 260, puts a very short machine language program into memory. This program is shown below:

```
0300- A9 0A      LDA  #$0A
0302- 85 67      STA  $67
0304- 4C 66 D5   JMP  $D566
```

The first two statements in this program place the value 10 (hex 0A) into location 103 (hex 67). Position 103 is the low byte (and position 104 is the high byte) of the starting location of the current BASIC program. The first statement in the program is "100 GOTO 200" and occupies 9 bytes: 2 bytes for the location of the next line, 2 bytes for the statement number, 1 byte for the GOTO token, 3 bytes for the digits of the number 200, and 1 byte for a terminating 0. Normally location 103 would contain the value 1, so adding 9 to this value makes the computer think that the BASIC program begins at the second line. To see that this works, enter the BASIC program and then POKE 103,10. If you list the program after this POKE, the list will begin with line 20. POKEing 103 with the value 1 will restore the program to begin with statement 10.

The final line in this machine language program jumps to 54630 (hex D566) where the RUN routine in firmware Applesoft BASIC resides. Since the value in location 103 (hex 67) has been changed, the RUN command executes the BASIC program starting at line 110. Once the program is running, the section that can only be accessed by the RESET key, the value in location 103 is changed back to its standard value so that the RUN command in line 190 will RUN the program starting with the first line of the BASIC program.

Many variations on this general scheme are possible. By making the RESET key RUN statements of BASIC code, changing the RESET key function becomes an easy adaptation to add to any BASIC program.

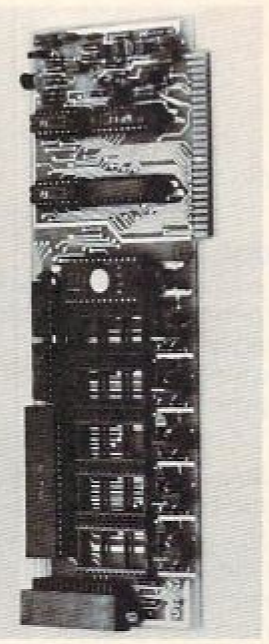
```
100 GOTO 200
110 REM **HERE IS WHERE THE RESET KEY
         SENDS THE COMPUTER**
120 POKE 103,1
130 CALL 1002
140 HOME
150 VTAB 3
160 PRINT "YOU HAVE PRESSED THE RESET
    KEY."
170 PRINT: PRINT "I WILL NOW RERUN THE
    PROGRAM."
180 FOR PAUSE=1 TO 2000:NEXT
190 RUN
200 REM **MAKE THE RESET KEY GOTO
         SECOND STATEMENT**
210 POKE 1010,0: POKE 1011,3:
    CALL -1169
220 REM -THE ABOVE STATEMENT RESETS
        "JUMP TO" LOCATION FOR RESET
230 DATA 169,10,133,103,76,102,213
240 FOR SPOT = 768 TO 774
250 READ CODE: POKE SPOT,CODE
260 NEXT
270 REM **PLACE BODY OF PROGRAM HERE**
280 HOME
290 VTAB 3
300 PRINT "THE PROGRAM IS NOW RUNNING."
310 GOTO 310
```

©

OMNI

The "Everything" I/O Board for the Apple II & II+

OMNI is a multi-function input/output board for the Apple II or II+ computer. It provides, on a single board, most of the "missing" features needed to make the Apple a complete computer. With OMNI your Apple can have:



- Parallel I/O with handshaking
- RS232 Level serial I/O (software driven)
- 24 Hr. Real Time Clock with Alarm
- 2K EPROM with graphics, I/O driver, and screen editor firmware
- 256 Byte PROM supervisory firmware
- Six 2K PROM/EPROM expansion sockets software selectable (one socket used for Eclectic firmware, 5 available for user)

INPUT:
- Generate full ASCII character set from keyboard
- Optional shift key detection
- User-definable "soft" keys with screen legends
- Integrated text line editor full cursor movement, insertion/deletion modes
- Key legend stickers included
- Demonstration Diskette with programming examples and a Soft Character Editor

OUTPUT:
- Full 96 character ASCII display
- Concurrent 64 user-definable "soft" characters
- Optional character overstrike and EOR on background
- Optional double-width color characters
- Character rotation in 90° steps

Never before have so many functions been available on a single board.

OMNI was designed with one major goal in mind, flexibility. The OMNI system consists of some extremely simple but very sophisticated hardware, a large amount of powerful firmware (programs permanently residing in Read Only Memory chips), and an equally extensive amount of software (programs residing on diskette that are loaded into RAM as needed). In addition, OMNI comes with extensive documentation.

ECLECTIC SYSTEMS CORPORATION

To order TOLL FREE: 1-800-527-3135
or Order by Mail Below

Eclectic Systems Corporation, P.O. Box 1166, Addison, TX 75001

Here's my order for OMNI at $268 plus $3 for shipping and handling (UPS surface, unless specified), $5 overseas.

☐ My check is enclosed.
☐ Please charge my VISA ____ Mastercard ____
Account # __________ Expires ________

Name ______________________________

Address ______________________________

City/State ______________ Zip __________

Signature ______________________________

A Simple Printer Interface For The Apple II

Marvin L. DeJong
Dept. of Math-Physics
The School of the Ozarks
Pt. Lookout, MO

In the January 1981 issue (**COMPUTE!** #8) I described a simple circuit that could be used with an Apple II to perform the experiments in my book$^{(1)}$. The circuit provided one eight-bit output port. These two ports can also be used to interface the Apple II to a parallel port printer.

The Circuit

For the unfortunates who do not have a copy of **COMPUTE!** #8, I have included the circuit diagram of the peripheral I/O card in this article. It is shown in Figure 1. My circuit was wire wrapped on a Vector Electronic #4609 plugboard which fits into the peripheral card connectors on the Apple II. For the purpose of this application, the eight LEDs and the DIP switch (with pull-up resistors) are not needed. They are only used if you wish to use the peripheral I/O board in conjunction with the experiments in my book. You may also wish to experiment with the possibility of omitting the 74LS242 bus transceivers and the associated logic, simplifying the circuit further. This would leave only the 74LS138, an inverter, the two 74LS75s, and the 81LS97. Since only one bit of the input port is used to interface to the printer, you may wish to replace the 81LS97 with a 74LS125. I used the circuit as it is shown in Figure 1, with the DIP switch removed from the socket.

My printer (which was not the one used to make the listings in this article) is a MICROTEK MT-80P which I normally use to interface to one of my TRS-80 machines. It claims to have a "Centronics-compatible interface," so perhaps the circuit and software we describe here may also be used with Centronics printers. The printer has eight data lines and several handshaking lines. The eighth bit is not used by the printer: it uses seven-bit ASCII. So seven bits of the output port on our peripheral I/O card are used to send the character to be printed to the MT-80P printer.

Two handshaking lines are used, DATA STROBE and BUSY. The microcomputer must supply a logic-zero pulse (strobe) of at least one microsecond in duration when the character on the data lines is to be read by the printer. Thus, the DATA STROBE line is controlled by the Apple II peripheral I/O card. In particular, I used bit zero (PA0 in Figure 1) to control the DATA STROBE line, while the seven-bit ASCII character appears on bits one to seven (PA1 - PA7). When the DATA STROBE pulse is sent, the printer responds by bringing the BUSY line to logic one. It stays at logic one until the character is read. This will only take about 40 microseconds unless the buffer is full. The BUSY line will stay high until there is room in the buffer. Thus, the BUSY line is connected to bit seven of the input port on the peripheral I/O card where it may be watched with a BMI instruction. Figure 2 shows the connections to the printer, and Figure 3 illustrates the handshaking sequence.

The Software

The machine language software driver routine is shown in Program 1. It is used with DOS 3.3, but other versions should work equally well. The machine language program consists of two parts. The first part starts with line six in the listing and ends with line 19 (locations \$02C0 – \$02DB). It has two functions:

1. It sets up the Apple II output registers (\$36 – \$37) to point to the printer routine at \$02E0, and it jumps to a DOS routine to fix the DOS output register. (See pages 103-104 in the APPLE II DOS MANUAL.)

2. It loads a form-feed character, \$0C, into the printer and pulses the DATA STROBE line.

The second part of the machine language routine is the actual print routine. It puts an ASCII character on the data lines to the printer and then it pulses the DATA STROBE line, but it does not do this unless the BUSY line is at logic zero, indicating that the printer is not busy. Finally, it jumps the monitor COUT routine that prints the character on the video monitor screen, before returning to the DOS program.

In Program 2 I show a greeting program that is the INITilization program on the slave diskette for our DOS 3.3. It gives the user the chance to call PRINTS, the object code file that is also stored on the slave diskette. This completes the description of the software for this system. Refer to the comments for more details regarding the software.

If you are not running a disk system, then to operate the printer load the machine language programs in Program 1 with a single modification. Replace the JMP DOSSYS instruction with a BRK.

Execute the program from the monitor, starting at $02C0. You can then either stay in the monitor or return to BASIC with a control B.

Notice that the software is located in page two of memory. If you type in a very long sentence you may wipe out your program, since it is part of the input buffer for the Apple II. Ideally, you would PROM the software. (We should add that the software as shown assumes that the peripheral I/O card is in *slot one* on the Apple II. The software, assuming the peripheral I/O card is in slot one, would be loaded into locations $C100 upward, *starting with the instructions at $02E0 in Program 1*.

To initialize the printer you would still want to execute instructions from $02C8 through $02D8, with a BRK replacing the JMP DOSSYS at location $02D9. Thereafter a PR#1 command would produce an active printer, and a PR#0 would disable it. I should add that I have *not* tried to run the system with the program in EPROM, but I think that I understand my Apple II enough to make the instructions just given. I would very much like to hear from someone who might try this approach.

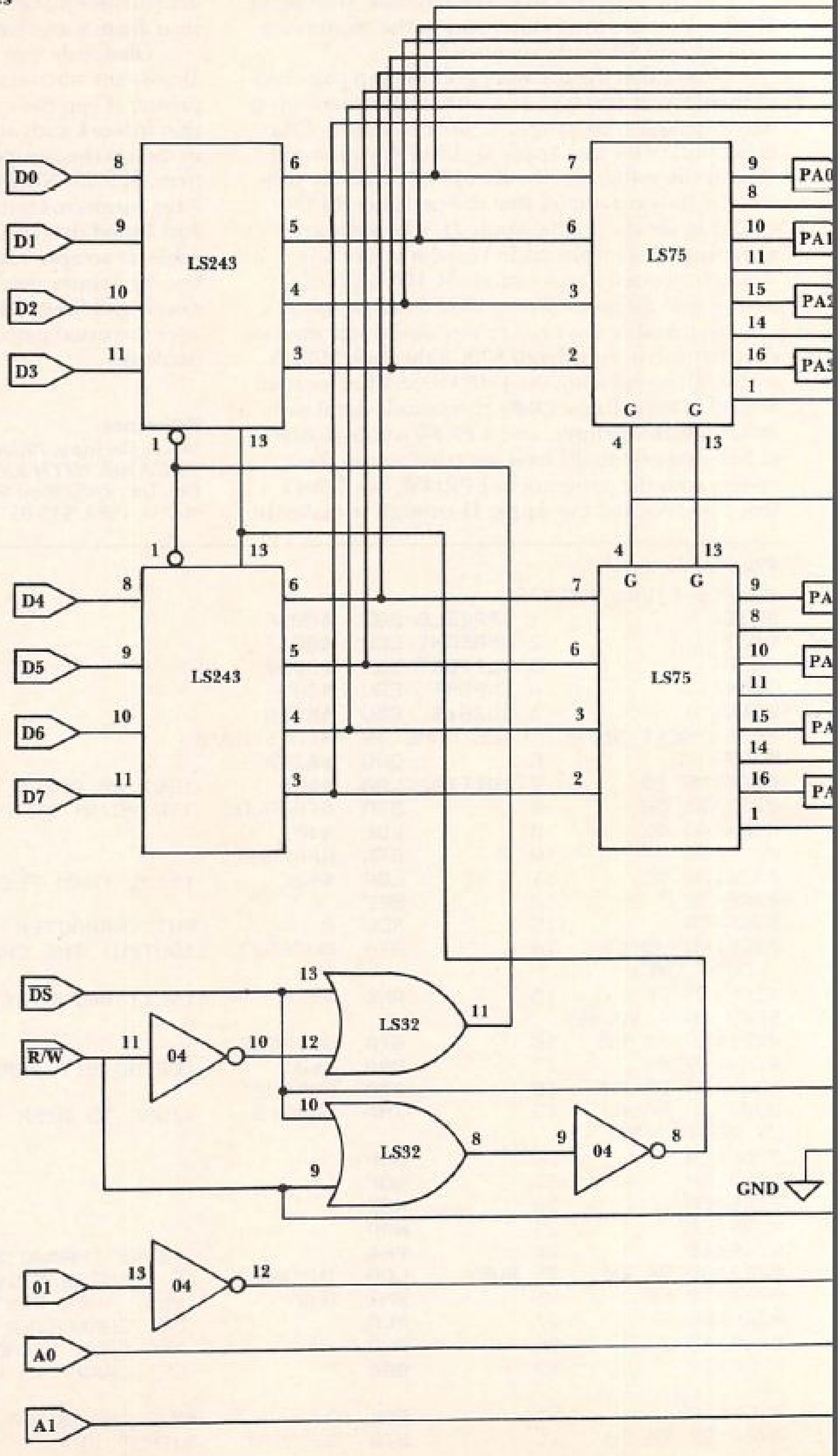
Obviously this interface circuit will work with almost any microcomputer system and any parallel printer. Even the software requires little modification to work with any 6502 based system. The card to mount the components is the most expensive item, $23.25. Note that the card I used has another edge connector not used to plug into the Apple II, and I used that connector to attach to our printer cable. It accepts a standard 20/40 edge connector, but my printer used a 19/36 edge connector, so I sawed and filed to fit. It has a big price advantage over the usual parallel interfaces found in your catalogs.

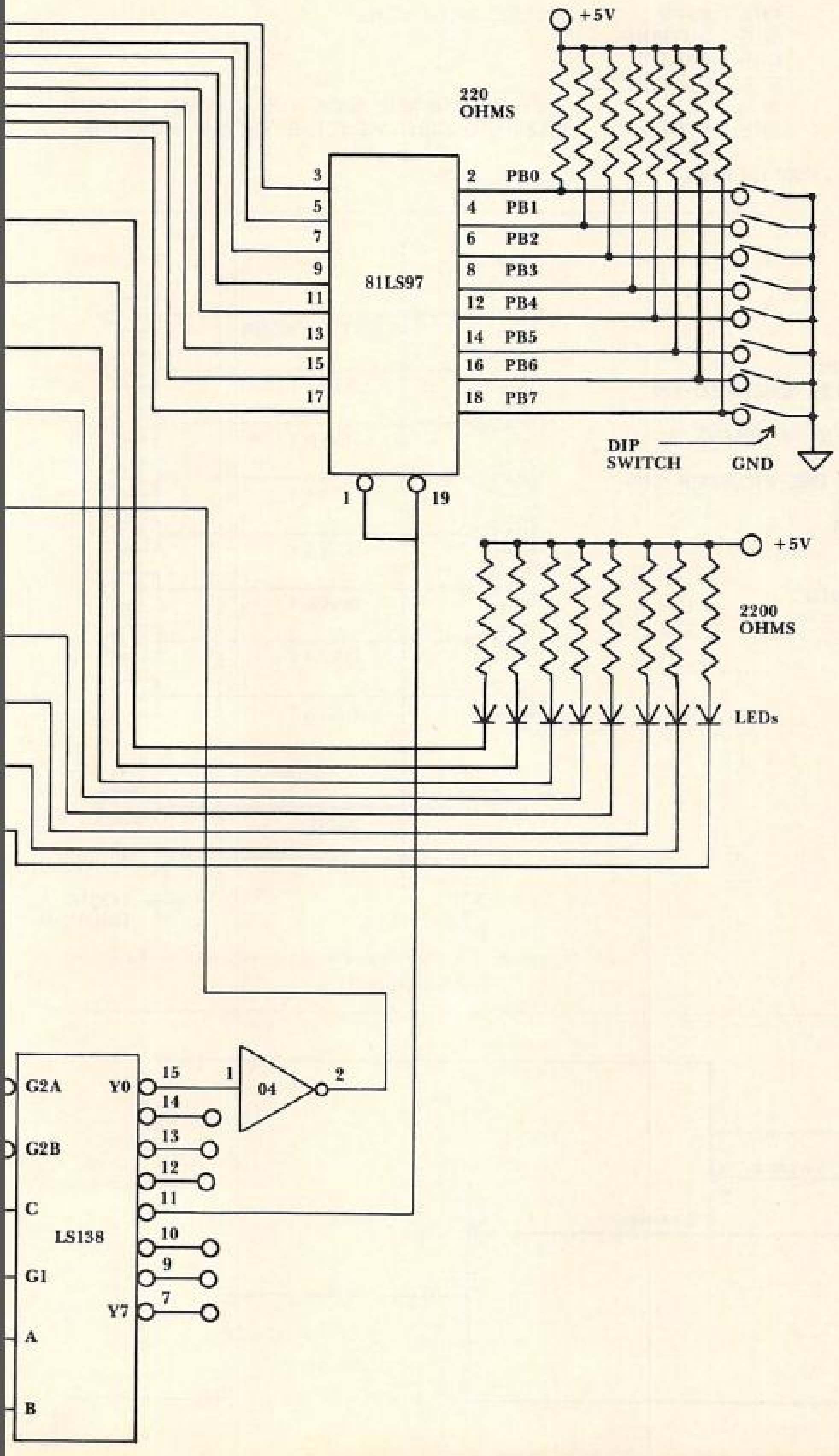
Reference
[1]M. L. De Jong, *PROGRAMMING & INTERFACING THE 6502, WITH EXPERIMENTS,* Howard W. Sams & Co., Inc. 4300 West 62nd St., Indianapolis, Indiana 46268, 1980. $15.95

Program 1.

```
SOURCE FILE: PRINTS
0036:             APREGLO EQU   $0036
0037:             APREGHI EQU   $0037
C090:             OUTPORT EQU   $C090
C094:             INPORT  EQU   $C094
03EA:             DOSSYS  EQU   $03EA
----- NEXT OBJECT FILE NAME IS PRINTS.OBJ0
02C0:                     ORG   $02C0
02C0:A9 E0        INITIAL LDA   #$E0        ;SET UP APPLE OUTPUT REGISTERS
02C2:85 36                STA   APREGLO     ;TO POINT TO PRINTER ROUTINE
02C4:A9 02                LDA   #$02
02C6:85 37                STA   APREGHI
02C8:A9 0C                LDA   #$0C        ;SEND FORM FEED TO PRINTER
02CA:38                   SEC
02CB:2A                   ROL   A           PUT CHARACTER IN HIGH ORDER 7 BITS
02CC:8D 90 C0             STA   OUTPORT     ;OUTPUT THE CHARACTER WITH BIT ZERO AT
 LOGIC ONE
02CF:29 FE                AND   #$FE        ;NEXT BRING BIT ZERO TO LOGIC ZERO TO
SEND DATA PULSE
02D1:8D 90 C0             STA   OUTPORT
02D4:09 01                ORA   #$01        ;BRING BIT ZERO TO LOGIC ONE AGAIN
02D6:8D 90 C0             STA   OUTPORT
02D9:4C EA 03             JMP   DOSSYS      ;JUMP TO DISK ROUTINE TO EXCHANGE OUTP
UT REGISTERS
02DC:EA                   NOP
02DD:EA                   NOP
02DE:EA                   NOP
02DF:EA                   NOP
02E0:48                   PHA                ;SAVE CHARACTER
02E1:AD 94 C0     BUSY    LDA   INPORT      IS PRINTER STILL BUSY?
02E4:30 FB                BMI   BUSY        ;YES, THEN DONT BOTHER IT
02E6:68                   PLA                GET CHARACTER BACK
02E7:48                   PHA                AND SAVE IT AGAIN
02E8:38                   SEC                SET CARRY TO ROTATE A ONE INTO BIT ZE
RO
02E9:2A                   ROL   A           MOVE CHARACTER UP ONE BIT
02EA:8D 90 C0             STA   OUTPORT     OUTPUT IT
```

Figure 1. Peripheral Card Data bus




+5V
220
OHMS
81LS97
3
5
7
9
11
13
15
17
2 PB0
4 PB1
6 PB2
8 PB3
12 PB4
14 PB5
16 PB6
18 PB7
DIP
SWITCH
GND
1
19
+5V
2200
OHMS
LEDs
G2A
G2B
C
LS138
G1
A
B
Y0
Y7
15
14
13
12
11
10
9
7
1
04
2

```
02ED:29 FE       32          AND  #$FE        ;PULSE DATA LINE
02EF:8D 90 C0    33          STA  OUTPORT
02F2:09 01       34          ORA  #$01
02F4:8D 90 C0    35          STA  OUTPORT
02F7:68          36          PLA               GET CHARACTER BACK FOR SCREEN OUTPUT
02F8:4C F0 FD    37          JMP  $FDF0       ;JUMP TO COUT ROUTINE IN THE MONITOR

*** SUCCESSFUL ASSEMBLY: NO ERRORS
```

Program 2.

```
]LISTLIST
?SYNTAX ERROR
]LIST

5  REM  GREETING PROGRAM
10  PRINT "SLAVE DISKETTE CREATED ON
    32K SYSTEM"
15  PRINT : PRINT : PRINT : PRINT :
    PRINT
21  INPUT "DO YOU WANT THE PRINTER ON?
    (TYPE Y OR N.) ";A$
22  IF A$ = "Y" THEN 30
23  GOTO 50
30 D$ =  CHR$ (4)
40  PRINT D$;"BRUN PRINTS"
50  END
```

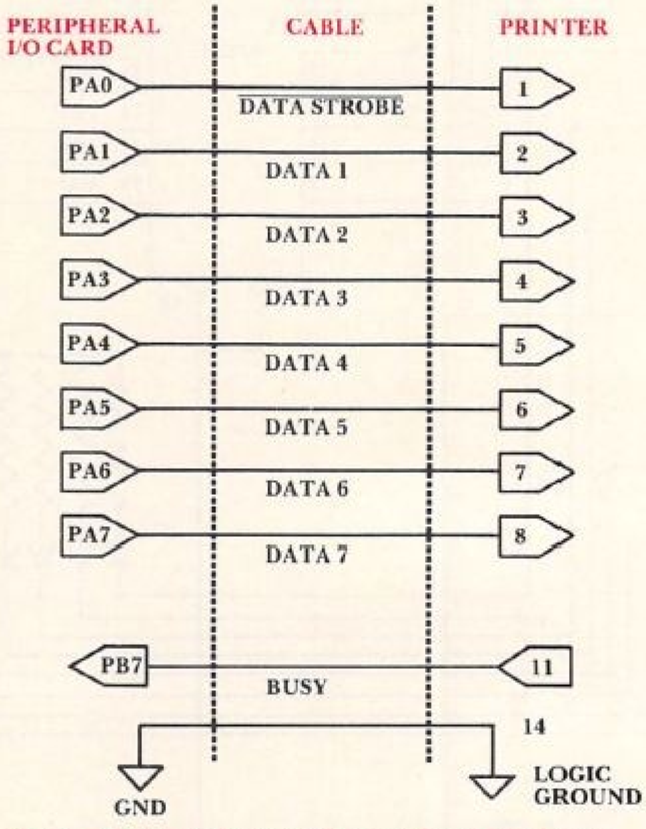


Figure 2. Interface between the peripheral I/O circuit and the printer.

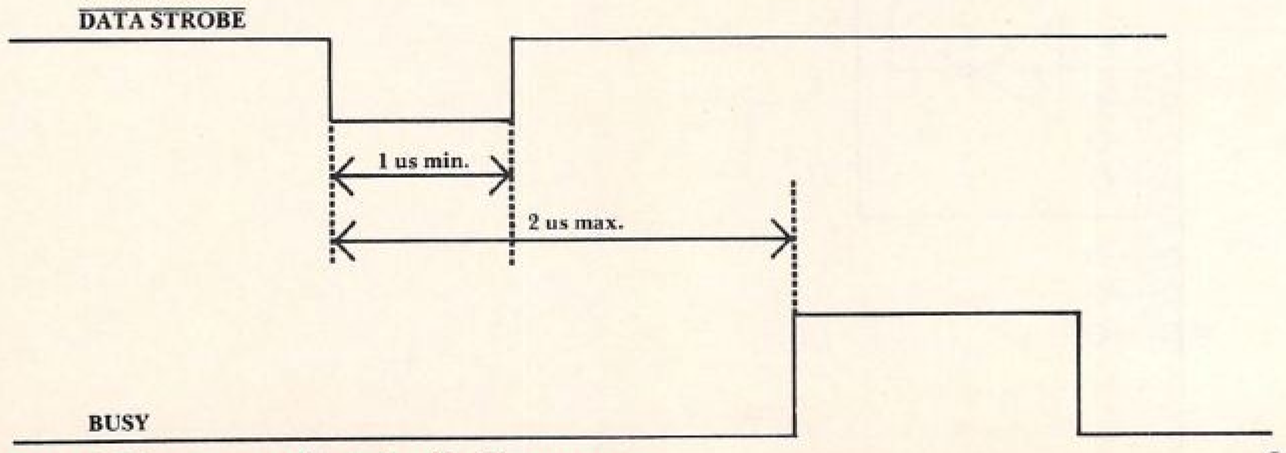


Figure 3. Microcomputer-Printer handshaking sequence.

APPLE II*
SOFTWARE FROM

POWERSOFT

PEGASUS

(a PASCAL based data base system)

PEGASUS—is a filing and retrieval system using the PASCAL programming language providing a general means for storing data in an orderly fashion. PASCAL code runs three to five times faster than BASIC code designed for a similar application.

Data stored in the PEGASUS data base may be modified, retrieved, and formatted into convenient reports. Three types of data are supported: character, real, and integer. Each PEGASUS data base record may contain up to 20 fields.

Data may be entered either interactively from the console or as a batch from a text file. Records may be modified after they have been entered or deleted from the data base entirely. PEGASUS may also be used to select groups of records based on the values of one or more fields. Output may be to the CRT screen, a printer or a text file. Thus, PEGASUS may be used to create printed reports, examine data on-line, or *interface with the input or output of other PASCAL programs.* Requirements: Apple II, Plus, or III and two 5¼" disk drives. Or an 8" or Winchester type drive. USCD Pascal Language System. . . . .
5¼" Disk Only/$199.95

INCOME STATEMENT SYSTEM

INCOME STATEMENT SYSTEM—(Summarized Reports including Budget Figures Based on Super Checkbook III transactions.)—An excellent program complement to SUPER CHECKBOOK III. The system provides for up to 100 income and expense codes. For each code the system maintains a total for the current month, current budget, current year-to-date, and three prior year-to-dates. Income codes may have up to six corresponding expense codes. A "sort code" feature allows account codes to print in a user defined sequence.

Updates to the accounts include current month, end-of-month, and end-of-year. Gross and Net Income Statements may be printed in either account code or sort code sequence. The Account Master File List may be printed by sort code, account code, or alphabetically by account name. Detailed transactions for each code are printed and totaled, one code per page, in code number order.

This system is designed to run in conjunction with the SUPER CHECKBOOK III program described below. Requirements: 48K, two disk drives, printer card, Applesoft
Disk Only/$49.95

SUPER CHECKBOOK III

SUPER CHECKBOOK III—A *vastly improved* version of our popular selling program. With new features such as: simplified but powerful transaction entry and modification routines, new reconciliation routines, additional features such as 30 percent increase in the total number of checks handled, posting of interest from interest bearing checking accounts, automatic teller transactions, bullet proof error handling, and smart disk routines. Plus the program still contains the options of bar graphs, sorting, activities, and account status. See INCOME STATEMENT SYSTEM described above.
Disk Only/Applesoft $49.95

Dealer Inquiries Invited

Visa and MasterCard, Check or Money Order include $2.00 for shipping and handling. C.O.D. $2.00 additional.

*Apple II and Applesoft are registered trademarks of APPLE COMPUTER INC.

POWERSOFT

P. O. BOX 157
PITMAN, NEW JERSEY 08071
(609) 589-5500

The Voice gives your Apple the power of speech! Use the standard Voice vocabulary to speak an endless combination of phrases, or easily record your own vocabulary and make your Apple say anything you like. Each data disk stores up to 80 words or phrases which can be sorted for quick reference. What's more, the Voice allows you to speak from any Basic program by using Print Commands. Guaranteed the best and easiest to use speech software available. For the Apple II or Apple II Plus with 48K. ($39.95)

from the leader in quality software

MUSE SOFTWARE™

Apple II is a trademark of Apple Computer Corp.

330 N. CHARLES STREET
BALTIMORE, MD 21201
(301) 659-7212

Call or write for information and the name of your nearest **MUSE** dealer

TELECOMMUNICATIONS PACKAGE FOR THE APPLE II™ FOR UNDER $300!

Package includes MICROCONNECTION™ with autodial/autoanswer module, stand-alone serial communications card with serial printer port and a terminal program. For more information, write or phone:

the microperipheral corporation

2643A · 151st Pl. N.E.
Redmond, WA 98052
(206) 881-7544

APPLE • ALTOS • ATARI • MAXELL • DYSAN • EPSON • CCS • SHARP • CASIO • HP • VERBATIM • MEMOREX • SOROC • CORVUS • ADDS

BMC • CENTRONIC • NEC • HAZELTINE • AMPEX • VERBATIM • HITACHI • HAYDEN • CANON • NOVATION • D.C. HAYS MICROCOMPUTERS • PAPER TIGER • PERSONAL SOFTWARE • GTE •

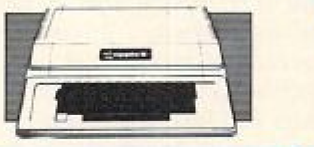
HP • D.C. HAYS MICROCOMPUTERS • PAPER TIGER • GTE • DIABLO • BMC • CENTRONIC • NEC • HAZELTINE • MEMOREX • AMPEX • PERSONAL SOFTWARE • ADDS • XEROX • DIABLO

MICRO BUSINESS WORLD INC.

Information Line (213) 996-2252
TOLL FREE MAIL ORDER LINES
(800) 423-5886 Outside Calif.

SPECIAL OF THE MONTH

APPLE II PLUS 48K W/16K EXPANSION BOARD = 64K



Our price $1199.00

List price $1780.00
Our price $1199.00
Save $581.00

COMPUTERS

apple computer

	List	Our Price	SAVE
Apple II-16K	$1330.00	$999.00	$331.00
Apple II-32K	1430.00	SPECIAL CALL	$
Apple II-48K	1530.00		$
Apple II-64K	1725.00		$
Apple III 128-K	3915.00	2997.00	918.00
ALTOS 8CS 8000-15	5990.00	4450.00	1540.00
SHARP-64K Z80 FULL KBD YX-3200			CALL
NEC PC 8001	1295.00	1050.00	245.00

ATARI PERSONAL COMPUTERS

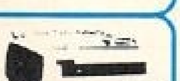
	List	Our Price	SAVE
400 16K Bytes of Ram	595.00	337.45	257.05
800 16K Bytes of Ram	$1080.00	739.00	332.00
410 Program Recorder	90.00	77.00	13.00
810 Disk Drive	600.00	457.00	143.00
825 Printer (80 col-Centronic 737)	999.95	769.00	230.00
820 Printer (40 col impact)	450.00	353.00	97.00
830 Acoustic Modem	200.00	155.00	45.00
850 Interface Module	220.00	192.00	48.00
Atari Visicalc	200.00	164.00	36.00
Atari 400-16K	595.00	327.00	304.00

HEWLETT PACKARD

	List	Our Price	SAVE
HP-125 Microcomputer	3750.00	2990.00	760.00
HP-85 Microcomputer	3250.00	2475.00	775.00
HP-83 Microcomputer	2250.00	1777.00	473.00
16K Exp-Memory Module	295.00	255.00	40.00
Graphics Plotter 7225	2450.00	2075.00	375.00
Personality Mod for 7225	750.00	665.00	85.00
2631B Impact printer hvy dty	3950.00	3250.00	700.00
Option 020 for 2631B	150.00	125.00	25.00
8 Drives to choose from 82902S	1300.00	1125.00	195.00
9895A 8" Dual Drive	6850.00	5500.00	1350.00
Graphics Tablet 9111A	2050.00	1676.00	374.00
HP-41 CV New 2.2 bytes mem	325.00	250.00	75.00
HP-41-C Calculator	250.00	185.00	65.00
Card Reader for 41CV/C	215.00	162.00	53.00
Printer for 41CV/C	385.00	289.00	101.00
Optical Wand for 41 CV/C	125.00	97.00	28.00
Quad Ram equals 4 Mem. Mods	95.00	81.00	14.00
Memory Modules for 41C		25.00	$
HP-97 Programble Printer	750.00	595.00	175.00
HP-67 Programble Calculator	375.00	295.00	80.00
HP-34C Programble Scientific	150.00	117.00	33.00
HP-38C Programble Bus. R/E	150.00	117.00	33.00
HP-32E Adv. Scientific	55.00	48.00	7.00
HP-37E Business Mgmt.	75.00	57.00	18.00

We carry a large inventory of Libraries, accessories and supplies.

PRINTERS



	List	Our Price	SAVE
EPSON			
MX 80 FT	745.00	549.00	196.00
MX 80 IMPACT	645.00	450.00	195.00
MX 70 IMPACT	500.00	390.00	110.00
MX 100	995.00	765.00	230.00
APPLE SILENTYPE	645.00	299.00	346.00
ANADEX 9501	1650.00	1299.00	351.00
NEC			
5510	3195.00	2445.00	750.00
5515	3295.00	2545.00	750.00
3510	2495.00	1795.00	700.00
3515	2545.00	1849.00	696.00
OKIDATA			
MICROLINE 80	545.00	395.00	150.00
MICROLINE 82	649.00	549.00	100.00
MICROLINE 83	1050.00	769.00	281.00
PAPER TIGER			
445G with Graphics	795.00	695.00	100.00
460G with Graphics	1,394.00	899.00	495.00
560G New full size	1,695.00	1,139.00	556.00
DIABLO (LETTER QUALITY)			
630 R102 bi-directional tractors	2,965.00	2,350.00	615.00
1640K109 keyboard tractors	4,000.00	2,899.00	1,100.05
630 RO Receive Only	2,710.00	2,250.00	460.00
1650K 136 keyboard tractors	4,000.00	3,100.00	900.00

APPLE II STUDENT SYSTEM

- Apple II Plus System-48K RAM
- Disk II Floppy Disk & Interface (DOS 3.3)
- 12" Grn. Phs. Video Monitor

SAVE $655

Our Price **$1795**

List $2450.00

MONITORS

	List	Our Price	SAVE
NEC Grn. Phs. 12"	$285.00	239.99	$46.00
BMC Green Phs. 12"	275.00	229.00	46.00
Zenith 12"	159.00	119.00	30.00

SANYO MONITORS
High Resolution, Number 1 seller

	List	Our Price	SAVE
13" Color (new) high quality	$550.00	388.00	$162.00
12" Green Phosphorous	360.00	240.00	120.00
12" Black & White	340.00	217.00	123.00
15" Black & White	370.00	235.00	135.00
9" Black & White The Best Seller!	235.00	145.00	90.00

DISKETTES

SOLD IN BOXES OF 10
(Min. Purchase)
BUY $100 of the Following Diskettes

	List	Our Price	SAVE
DYSAN	PRICE PER DISKETTE		
104/1 5" SOFT SECTOR	$ 6.00	3.99	$ 2.01
104/1D = " DBL DEN-SOFT SEC	6.40	4.60	1.80
3740/1 8" SOFT SECTOR	7.25	4.75	2.50
3740/1D 8" DBL DEN-SOFT SECTOR	10.75	6.90	3.85
MAXELL			
MD-1 5" SOFT SECTOR/DBL. DEN.	5.00	4.00	1.00
MD-2 = " SOFT SECTOR/DBL. SIDE/DBL. DEN.	7.00	4.90	2.10
FD-1 8" SOFT SEC/DBL DEN.	6.50	4.50	2.00
FD-2 8" SOFT SEC/DBL SIDE/DBL. DEN.	8.50	5.95	2.55

TERMINALS

TELEVIDEO	List	Our Price	SAVE
910	699.00	599.00	100.00
912C	950.00	699.00	251.00
920C	995.00	795.00	200.00
950C	1,195.00	949.00	246.00

DRIVES

CORVUS	List	Our Price	SAVE
5 MBYTES	3,750.00	3,050.00	700.00
10 MBYTES	5,350.00	4,449.00	901.00
20 MBYTES	6,450.00	5325.00	1,125.00

SOFTWARE

FOR APPLE II

	List	Our Price	SAVE
Apple Fortran	$ 200.00	$ 147.00	$ 53.00
DOS 3.3 (16 Sector)	60.00	45.00	15.00
Apple PILOT (16 Sector)	150.00	119.00	31.00
Apple FORTRAN (requires A2B0006) (16 Sector)	200.00	159.00	41.00
Language System with Apple Pascal	495.00	399.00	96.00
BPI General Ledger System	395.00	319.00	76.00
BPI Inventory Package	395.00	319.00	76.00
Visidex	200.00	159.00	41.00
Visicalc	200.00	159.00	41.00
Desktop Plan II	200.00	159.00	41.00
Microlab Database System	150.00	119.00	31.00
Stoneware DB Master	229.00	189.00	40.00
Muse Supertext II	150.00	119.00	31.00
Softape Magic Window	99.00	72.00	27.00

TEXAS INSTRUMENTS

	List	Our Pr	SAVE
TI 99/4A Console New Improved	950.00	385.95	564.05
10" Color Monitor High Resolution	374.95	317.95	57.00
32K Memory Module	399.95	312.95	87.00
Speech Synthesizer	149.95	127.95	22.00
Disk Memory Drive	499.95	390.95	109.00
RF Modulator	49.95	42.50	7.45
Telephone Coupler (Modem)	224.95	185.95	39.00
Printer (Solid State)	399.95	315.95	84.00

MODEMS

	List	Our Price	SAVE
NOVATION CAT MODEM	189.95	140.00	49.95
NOVATION D-CAT	199.00	150.00	49.00
NOVATION APPLE-CAT	349.00	310.00	39.00
HAYES MICROMODEM	379.00	295.00	84.00
HAYES SMARTMODEM	279.00	239.00	40.00

CALCULATORS

CASIO	List	Our Price	SAVE
HR-10	49.95	39.00	10.95
HR-12	54.95	42.00	12.95
FR-100	79.95	59.00	20.95
FR-1210	129.95	99.00	30.95
PQ-20	29.95	23.00	6.95
LC-785	12.95	10.00	2.95
LC-3165	12.95	10.00	2.95
FX-68 Scientific	29.95	23.00	6.95
FX-81 Scientific	19.95	17.00	2.95
FX-3600P Scientific	39.95	29.95	10.00
FX-602P "Computer Talk" 88 Memories Programming Upper & Lower Case Dot Matrix	129.95	99.00	30.95
FX-702P Solves Problems with Alpha-Numeric Clarity, uses Basic Language	199.95	159.00	40.95

TELE. ANSW DEVICES

PHONE MATE	List	Our Pr	SAVE
900	119.95	86.00	33.95
905 Remote	199.95	144.00	55.95
910	159.95	115.00	44.95
920	199.95	144.00	55.95
925 Remote	239.95	173.00	66.95
930 Remote	299.95	216.00	83.95
950 Remote	339.95	245.00	94.95
960 Remote	399.95	288.00	111.95

1 Year Extended WARRANTY
$99.00
[illegible]
INQUIRE

$AVINGS
$ERVICE
$ELECTION
$ATISFACTION

MEMBER [illegible]

MICRO BUSINESS WORLD
MAIL ORDER WAREHOUSE
18720 Oxnard, #108 Tarzana, CA 91356

Telex: 182852
Answer: MICKOTZNA

OUTSIDE CA CALL TOLL FREE 1 (800) 423-5886 IN CA (213) 996-2252

Name (Please Print) ____________________
Address ____________________
City ________ State ________ Zip ________

Qty.	Make	Model	Description	Price	Total

TOTAL ORDER $ ________
TAX IF APPLICABLE* ________
SHIPPING & HANDLING** ________
TOTAL ENCLOSED $ ________

*California residents add 6% sales tax.
**Add 3% for Shipping & Handling – [illegible]

Certified Check or M.O.
Bank Wire Transfer

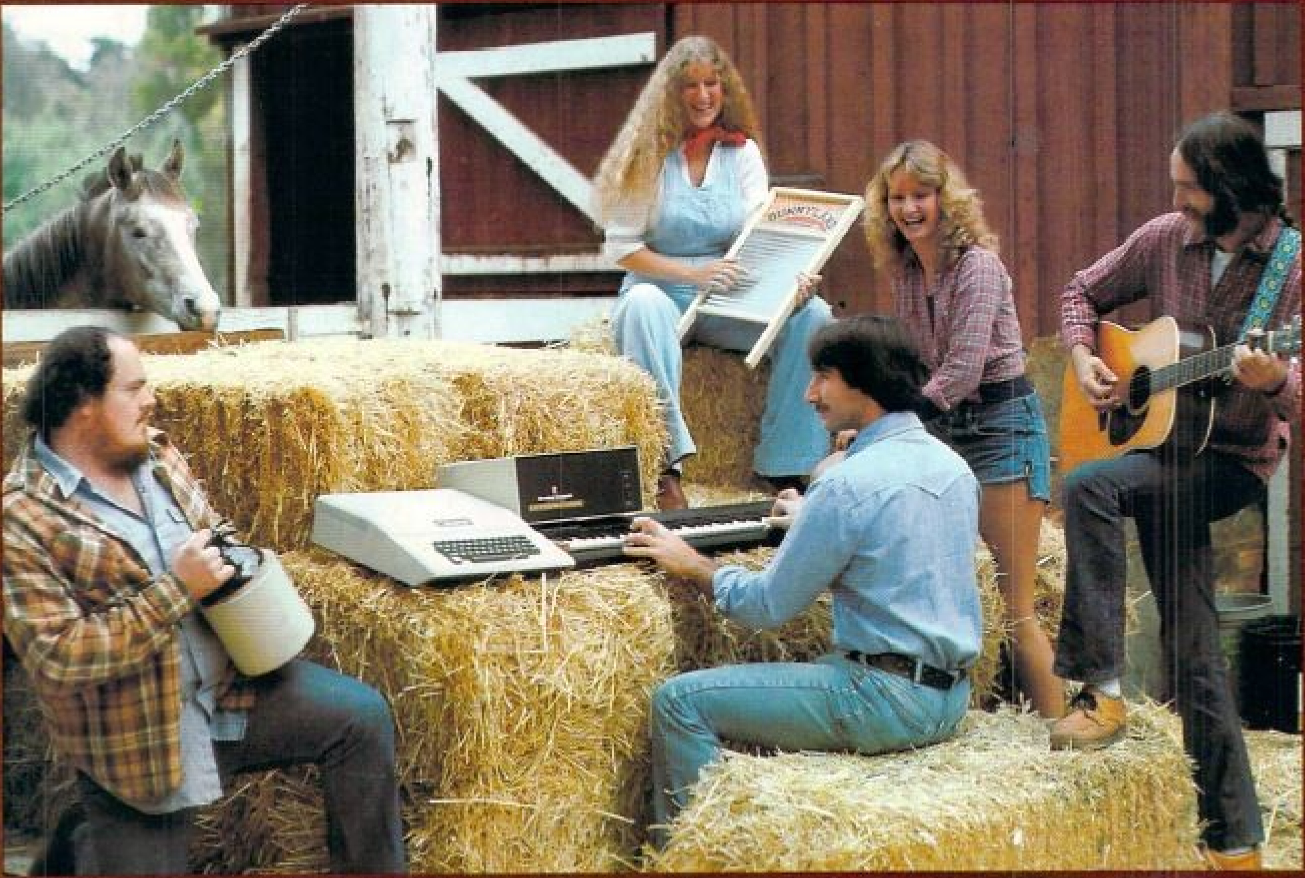
Cash/Credit Card (Allow 2 weeks clearance for checks)

Credit Card # ____________________
Exp. Date ________ Signature ____________________

We reserve the right to correct typographical errors. This ad supercedes all previous ads.

MAXELL • DYSAN • EPSON • CCS • SHARP • CASIO • HP • VERBATIM • MEMOREX • SOROC • CORVUS • PERSONAL SOFTWARE • CCS

MOUNTAIN MUSIC AIN'T WHAT IT USE TO BE!



MUSICSYSTEM™ FROM MOUNTAIN COMPUTER.

MusicSystem generates the sound of any musical instrument—real or imagined! Solo or sextet. Rock or classical. Laid-back or loud. At home, the concert hall, or the classroom. MusicSystem sets new standards for computer generated music.

- Works in your Apple®.
- Digital Synthesizer with 16 voices.
- Stereo output.
- Polyphonic—multivoice chords and note sequences.
- Additive synthesis of instruments.
- Waveforms are fully programmable for each voice to create instrument definitions and music dynamics.
- 35Khz sample rate.
- Frequency resolution is .5Hz steps.
- Graphical input of sheet music with a graphics printer.
- Complete software operating system.
- Graphical music editor using light pen (provided), game paddles, or keyboard.
- Pre-entered music provided for immediate playing and enjoyment.
- Thorough documentation and tutorial user's manual.

Drop by your Apple® Dealer and ask to hear "Mountain Music" for yourself.

NOW ONLY $395.00

suggested retail

Mountain Computer
INCORPORATED

300 El Pueblo, Scotts Valley, CA 95066
TWX: 910 598-4504 (408)438-6650

TASC™. The Applesoft* Compiler. It turns your Apple into a power tool.

Step up to speed. TASC, the Applesoft Compiler, converts a standard Applesoft BASIC program into super-fast machine code. By increasing program execution speed up to 20 times, Microsoft gives you a power tool for Applesoft BASIC programming.

Highest capacity available. TASC will compile and run larger programs than any other Applesoft Compiler. As a disk-based system, it doesn't require the simultaneous presence of compiler and program in memory. The memory you save allows you to compile significantly bigger programs.

Power without bulk. Code expansion of up to 100% severely restricts other compilers. TASC's special code compression schemes typically limit code expansion to only 25%. You'll really appreciate that with complex programs or programs that utilize Apple's hi-res graphic pages.

More BASIC power. TASC's powerful new commands increase Applesoft BASIC programming capability. Chain with COMMON allows compiled programs to share variables, so a main menu supports several programs in a single runtime environment.

TASC's True Integer Arithmetic and Integer FOR...NEXT capabilities maximize the execution speed of compiled programs. TASC's near total compatibility with Applesoft speeds compilation of existing programs with little or no modification.

What about mistakes? You perfect your programs interactively with Applesoft. If something does slip by, TASC recovers from errors discovered in compilation and traps all runtime errors. It even permits graceful interruptions during compilation.

See for yourself. Ask for a demonstration of TASC at your Microsoft dealer. Discover the software package that turns your Apple into a power tool.

MICROSOFT
CONSUMER PRODUCTS
A Division of Microsoft Inc.
10700 Northup Way • Bellevue, WA 98004

Apple Addresses

Bill Grimm
Mountain View, CA

The Apple II uses three types of addressing depending upon the language being used. Apple's machine language uses hexadecimal addresses in the range from $0000 to $FFFF. Its Floating Point BASIC language uses decimal addresses in the range from 0 to 65535. Its Integer BASIC uses decimal addresses in the range from 0 to 32767 to -32767 to -1. This means that, if you want to address a particular memory location, you must choose the correct address for the language you are using. Since I program in all three languages and my references are a mixture from all three, I needed an address cross-reference program. So I wrote "Apple Addresses."

"Apple Addresses" can be used "as is" to convert one language's address to another's, and to give the high and low byte values which need to be POKEd into a BASIC program to store that address. Alternatively, you could extract the subroutines in Apple Addresses which convert between hex and decimal numbers and insert them in your own program. See the last paragraph of this article for more details.

The program begins by asking the user which of the six possible conversions he would like to make. This is followed by a request to select the way the results of the conversions are to be displayed. There are four possible displays:

1. single conversions displayed on the monitor one at a time.

2. single conversions printed out on a Silentype printer* one at a time.

3. a range of conversions displayed on the monitor.

4. a range of conversions printed out on a Silentype printer*.

*With slight program modifications other printers could be used.

Subroutines

"Apple Addresses" makes extensive use of subroutines. This helps in organizing the program as well as making it shorter and easier to debug. The controlling or EXECutive routine is called Apple Addresses – Exec. It starts on line 100 and goes to line 310. Since a picture is worth a thousand words, I made what I call a *balloon diagram* (Figure 1) to show how data flows through the program. These are the conventions I used to make the diagram;

1. Each balloon represents a subroutine. The name of the subroutine and the line numbers where it is located are placed in the balloon.

2. Data flows through a subroutine in the direction of the arrows on the outside of the balloon.

3. Data flows between subroutines in the direction of the arrows on the *strings*.

4. If conditions are placed on what data flows through a subroutine, these conditions are written in along the *strings*.

As an additional aid for understanding how the program works I have included the following variable descriptions list:

A( ) — each A(I) holds the decimal equivalent value of the Ith hexadecimal numeral in the hex number being created from a decimal number — appropriate numbers are then added to convert these to ASCII codes.

A$( ) — holds the characters represented by the ASCII codes in A( ).

CHOICE — holds the number of the conversion chosen — see lines 120 to 178.

DVL — holds the decimal value of the number being converted — may be either FP or INT decimal.

DVL$ — is the string equivalent of DVL and is used in the output routines.

FLAG — if FLAG = 1 then an invalid number was entered and the program returns to get a new number.

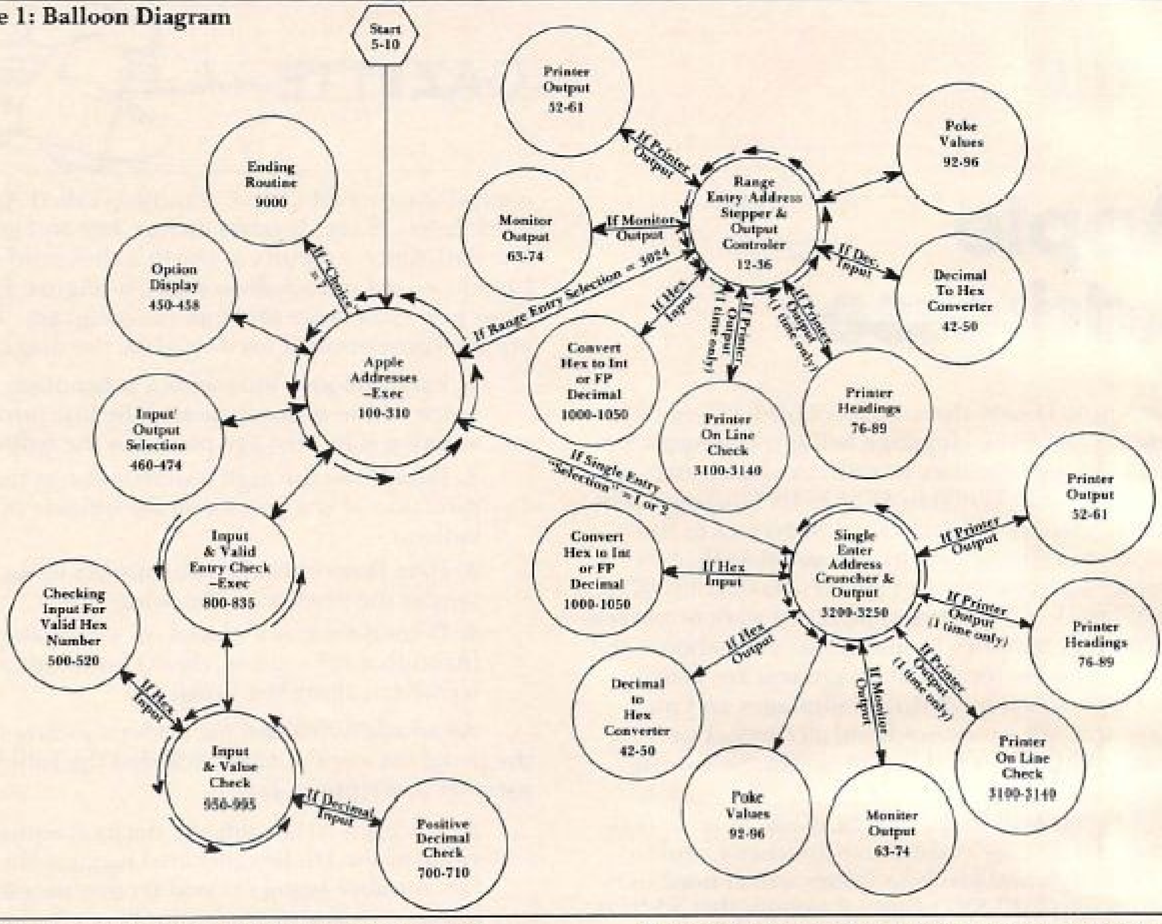
FRST — holds the FP BASIC address equivalent of the lowest address in the selected range.

FRST$ — holds the smallest address chosen — this address is then processed and stored in FRST.

HVL$ — holds the hex number selected or the hex number resulting from the conversion — if no hex numbers are involved then it holds the converted decimal number.

LST — holds the FP BASIC address equivalent

Figure 1: Balloon Diagram


Start 5-10
Ending Routine 9000
Option Display 450-458
Apple Addresses –Exec 100-310
Input/ Output Selection 460-474
Input & Valid Entry Check –Exec 800-835
Checking Input For Valid Hex Number 500-520
Input & Value Check 950-995
Positive Decimal Check 700-710
If Choice = 7
If Range Entry Selection = 3024
If Single Entry "Selection" = 1 or 2
If Hex Input
If Decimal Input
Printer Output 52-61
Monitor Output 63-74
Range Entry Address Stepper & Output Controler 12-36
Poke Values 92-96
Decimal To Hex Converter 42-50
Convert Hex to Int or FP Decimal 1000-1050
Printer On Line Check 3100-3140
Printer Headings 76-89
If Printer Output
If Monitor Output
If Dec. Input
If Hex Input
If Printer Output (1 time only)
If Printer Output (1 time only)
Single Enter Address Cruncher & Output 3200-3250
Printer Output 52-61
Convert Hex to Int or FP Decimal 1000-1050
Printer Headings 76-89
Decimal to Hex Converter 42-50
Printer On Line Check 3100-3140
Poke Values 92-96
Monitor Output 63-74
If Printer Output
If Hex Input
If Printer Output (1 time only)
If Hex Output
If Monitor Output
If Printer Output (1 time only)

of the largest address in the selected range.

LST$ — holds the largest address chosen — this address is then processed and stored in LST.

N — holds the decimal equivalent of each hex numeral in a hex number being converted to a decimal number.

PHI% — holds the number that would be POKEd into the high byte when placing the address into memory.

PLO% — holds the number that would be POKEd into the low byte when placing the address into memory.

POK — holds the address from which PLO% and PHI% are derived.

SELECT — holds the type of output selected — see lines 462 to 470.

STP — holds the positive decimal stepping interval chosen.

STP$ — holds the stepping interval chosen which is later changed and stored in STP.

TB — the horizontal tab value desired.

TN — holds the intermediate numbers of the decimal address that is being converted into a hex address. VTB — used to control the vertical tabbing of the monitor output.

Some Suggestions

I have found that the easiest way to debug a program while I am entering it is to first type in the EXEC program. Then, if I place return statements at all the branching locations, I can check the EXEC for bugs. Once the EXEC is free of bugs, I add one subroutine at a time in the order that the EXEC uses them, checking for bugs as I go.

If you have a need for subroutines which convert numbers from hex to decimal or from decimal to hex, two subroutines in this program may be of help. The first is called "decimal to hex converter" (lines 42 to 50). The input to this routine is TN which must hold a positive decimal number <65536. The output is HVL$ which holds the hex equivalent to the number in TN. The second is called "convert hex to INT or FP decimal" (lines 1000 to 1050). The input to this routine is HVL$ which must hold a hex number <= $FFFF and choice. If choice = 1 then you get the positive decimal equivalent. Otherwise you get Int BASIC's equivalent. The output is a decimal number in DVL. ©

More Apple Hi-Res Shape Writer

Chris Dupuy
Gonzales, LA

Countless hours spent plugging ones and zeros on graph paper are now history, thanks to Mr. Hennig's "Hi-Res Shape Writer." [**COMPUTE! #14**] Shload miseries are not missed and drawing shapes other than right angles are now a breeze.

After creating one star cruiser after another, I was soon struck with the harsh realization that I could not SAVE these cosmic creations on my cassette recorder. Unfortunately for me, I belong to the one percent club of Apple owners who cannot afford the luxuries of a disk drive. Undaunted with the PEEKs and POKEs ahead of me, I proceeded to write a routine that would put all the bytes from the shape table into trusty DATA statements.

The program is intended to be added to the original "Apple Hi-Res Shape Writer" by Doug Hennig. However, the routine used to POKE DATA in DATA statements can be adapted to other programs where the user does not want to be bothered with the rules of STOREing and RECALLing arrays.

Program Operation

REMark statements were omitted from the program in order to save valuable space, since memory size becomes a problem with complex shapes.

5–1084 Sets an array to the bytes POKEd into the shape table in original program.

13900–13906 Searches for the memory locations of the first blank DATA statement and sets Y equal to this.

13910–13970 POKEs Q to first item in DATA statement.

13930 Separates Q into individual digits.

13975 POKEs number of shape tables and reference numbers for shape tables.

14000–14075 POKEs bytes of shape table into the succeeding locations of the DATA statements.

14004 Searches DATA statement for a period (CHR$(46)), in order to find location to insert next value.

14550–14630 Demonstration program to verify information in DATA statements.

14572 Checks DATA statement to verify additional space on current statement. If not, then READ asterisks and jump to next DATA statement.

14700–14710 DELetes main portion of program and leaves demo program with DATA statements to be SAVEd.

15000–15005 DATA statements with 184 periods (quantity is at your discretion), and 4 asterisks.

Variables Used

Q Holds the number of bytes in the shape table.

V() Stores individual bytes of shape table.

Y Keeps track of the DATA statement memory locations.

R Used to check memory locations for a period.

F,FF Holds LENgth of strings and uses that value in FOR-NEXT statements.

T(),L() Arrays that hold the individual digits of bytes from shape table.

E$ User input.

X The location for bytes to be POKEd into shape table.

Y$ Stores the DATA being READ from demo program. String is used to prevent error message when asterisk is READ.

Hints And Changes

Those who have 32K Apples will encounter space problems when trying to run this longer program. DELeting the instructions, REMarks, disks subroutines, and combining statements will help avoid this obstacle.

Once all changes are made to your program, lines 13904 and 13906 may be DELeted. However, the memory location for the first DATA statement must be found. In machine language, the three bytes to look for are: 83 00 2E. The decimal location of 2E should then be set to Y in line 13900. Remember — if this change is done, no other changes can be made in the program (except for DATA statements), without the information being POKEd into the wrong locations. If searching for memory locations is too tedious, then you might want to experiment by raising the value in line 13900. Either one of these changes will save time in program execution.

Providing you have shaved off a good portion of the program, the value in line five may be raised to accomodate more complex shapes.

The major shortcoming in this program is the

SAVE/ on Software for APPLE II

SUPER SPECIAL – 50% OFF ON THE FOLLOWING APPLE PROGRAMS

APEX—The assembly language operating system for the Apple II—Total control of your Apple—fast program loading and execution—Fexibility—Efficiency.
Now only $50.00

XPLO—A Block Structured—Pascal type high level language operating under APEX—
Now only $40.00

HANDY DISK—Utility programs and device handlers for the APEX operating system—A powerful alternative to Apple DOS.
Now Only $20.00

THE EXTENDER—Extends Applesoft™ by adding these functions:
● Print using ● Auto Line Numbering ● Decimal/Hex Conversion ● High Resolution Screen Color Indication ● Single Color Screen Fill and more...
Now Only $25.00

MORE APPLE SPECIALS

PASCAL GENERAL LEDGER—Menu driven general ledger program based on the Osborne System. Customized utility for a true turn-key general ledger system.
$149.00

PROM BLASTER—Programs all 1K to 4K PROMS of 25XX & 27XX single or multi-voltage types—With all personality modules and read/write software.
$149.00

PROFESSIONAL FOOTBALL PREDICTIONS

$29.95

- Makes predictions for all 14 NFL games each week.
- Program's data base is updated weekly from your local newspaper's game statistics.
- Keeps complete record of all final scores and standings.
- Re-useable year after year.
- Very sophisticated program, yet easy to operate.

FOR YOUR APPLE II

*NOTE: Start saving your game results at the time of your order, APPARAT will provide the data base for the previous weeks games on the disk.

MISCELLANEOUS SUPPLIES

DISKETTES

Double density, soft sectored, replacement guaranteed, Spindle/Hub protected. (5¼" only)

Verbatim Datalife 5¼" 40 track	$24.95

PAPER

8½"x11" blank white, tractor feed paper, full box	$24.95
8½"x11" blank white, tractor feed paper, half box	$14.95
14½"x11" green bar, tractor feed paper, full box	$34.95
3½"x15/16" tractor feed mailing labels	$19.95

OTHER

5¼" plastic library case	$ 1.95
5¼" Flip-sort	$21.95
16K memory kits	$19.95

APPARAT'S TOP 15 APPLE GAMES

	SALE PRICE
1. Castle Wolfenstein (Muse)	$24.95
2. Gorgon (Sirius)	33.95
3. Raster Blaster (Budgeco)	24.95
4. Snoggle (Broderbund)	20.95
5. Robotwar (Muse)	33.95
6. Alien Typhoon (Brod.)	24.95
7. Space Eggs (Sirius)	24.95
8. ABM (Muse)	20.95
9. Ultima (Ca. Pacific)	33.95
10. Pool 1.5 (IDS)	29.95
11. Voyage of the Valkyrie (AOS)	33.95
12. Zork (Personal Software)	33.95
13. Shuffleboard (IDS)	24.95
14. Olympic Decathlon (Microsoft)	24.95
15. Apple Panic (Brod.)	24.95

Apparat, Inc. "On-going Support for Microcomputers"

4401 S. Tamarac Pkwy. • Denver, CO 80237 • (303) 741-1778 • (800) 525-7674

Freight F O B Denver call for shipping charges Foreign Orders shipped Air Freight inability to store more than one shape table at a time. Though a small amount of effort could change this, it would not be feasible if you are running low on memory. I hope this program brought some relief and enjoyment to you cassette owners out there.

```
5  DIM V(250)
1082  Q=Q+1
1084  V(Q)=X
13040  TEXT: HOME
13042  VTAB 10: HTAB 5
13045  PRINT "MEMORY LOCATIONS ARE BEING
       SCANNED"
13900  Y=3500
13904  IF PEEK(Y)=131 AND PEEK (Y+2)=46 THEN
       Y=Y+2: GOTO 13910
13906  Y=Y+1:GOTO 13904
13910  FF=LEN(STR$(Q))
13920  FOR X=1 TO FF
13930  T(X)=VAL(MID$(STR$(Q),X,1))
13940  POKE Y,T(X)+48
13945  Y=Y+1
13950  NEXT
13970  POKE Y,44
13975  POKE Y+1,49:POKE Y+2,44:POKE Y+3,48:POKE
       Y+4,44:POKE Y+5,52:POKE Y+6,44:POKE
       Y+7,48:POKE Y+8,44
13997  TEXT: HOME
13998  VTAB 10: HTAB 2
13999  PRINT "DATA IS NOW BEING POKED INTO
       MEMORY"
14000  FOR QQ=1 TO Q
14003  R=PEEK (Y)
14004  IF R<>46 THEN Y=Y+1: GOTO 14003
14005  F=LEN(STR$(V(QQ)))
14010  FOR T=1 TO F
14019  L(T)=VAL(MID$(STR$(V(QQ)),T,1))
14040  POKE Y,L(T)+48
14050  Y=Y+1
14055  NEXT
14060  POKE Y,44
14070  Y=Y+1
14075  NEXT
14100  HOME
14500  PRINT "TYPE 'ESC' KEY TO DEMONSTRATE
       PROGRAM"
14510  GET E$: IF E$<>CHR$(27) THEN END
14550  POKE 232,0: POKE 244,64
14555  READ Q
14560  FOR X=16384 TO 16387+Q
14570  READ Y$
14572  IF Y$="*" OR Y$="**" OR Y$="***" OR Y$=
       "****" THEN 14570
14575  Y=VAL(Y$)
14580  POKE X,Y
14590  NEXT
14600  POKE 16388+Q,0
14610  HGR: SCALE=1: ROT=0
14620  HCOLOR=3
14630  DRAW 1 AT 140,80
14700  VTAB 22
14702  PRINT "TYPE 'ESC' TO FORM NEW PROGRAM"
14704  GET E$: IF E$<>CHR$(27) THEN END
14705  TEXT: HOME
14706  PRINT "PROGRAM IS NOW READY TO BE
       SAVED"
14710  DEL 5,14510
15000  DATA . . . . . . . . . . . . . . . . . . . . .
           . . . . . . . . . . . . . . . . . . . . . . . .
           . . . . . . . . . . . . . . . . . . . . . . . .
           . . . . . . . . . . . . . . . . . . . . ****
15001  DATA . . . . . . . . . . . . . . . . . . . . .
           . . . . . . . . . . . . . . . . . . . . . . . .
           . . . . . . . . . . . . . . . . . . . . . . . .
           . . . . . . . . . . . . . . . . . . . . ****
15001  DATA . . . . . . . . . . . . . . . . . . . . .
           . . . . . . . . . . . . . . . . . . . . . . . .
           . . . . . . . . . . . . . . . . . . . . . . . .
           . . . . . . . . . . . . . . . . . . . . ****
15002  DATA . . . . . . . . . . . . . . . . . . . . .
           . . . . . . . . . . . . . . . . . . . . . . . .
           . . . . . . . . . . . . . . . . . . . . . . . .
           . . . . . . . . . . . . . . . . . . . . ****
15003  DATA . . . . . . . . . . . . . . . . . . . . .
           . . . . . . . . . . . . . . . . . . . . . . . .
           . . . . . . . . . . . . . . . . . . . . . . . .
           . . . . . . . . . . . . . . . . . . . . ****
15004  DATA . . . . . . . . . . . . . . . . . . . . .
           . . . . . . . . . . . . . . . . . . . . . . . .
           . . . . . . . . . . . . . . . . . . . . . . . .
           . . . . . . . . . . . . . . . . . . . . ****
15005  DATA . . . . . . . . . . . . . . . . . . . . .
           . . . . . . . . . . . . . . . . . . . . . . . .
           . . . . . . . . . . . . . . . . . . . . . . . .
           . . . . . . . . . . . . . . . . . . . . ****
```

©

The Voice gives your Apple the power of speech! Use the standard Voice vocabulary to speak an endless combination of phrases, or easily record your own vocabulary and make your Apple say anything you like. Each data disk stores up to 80 words or phrases which can be sorted for quick reference. What's more, the Voice allows you to speak from any Basic program by using Print Commands. Guaranteed the best and easiest to use speech software available. For the Apple II or Apple II Plus with 48K. ($39.95)

from the leader in quality software

MUSE SOFTWARE™

Apple II is a trademark of Apple Computer Corp.

330 N. CHARLES STREET
BALTIMORE, MD 21201
(301) 659-7212

Call or write for information and the name of your nearest **MUSE** dealer

Lower Case With Unmodified Apple

Joseph Wrubel
Aberdeen, NJ

This article describes a program called LC.EDIT which can be used to build, modify, and print text files using both upper and lower case letters on an unmodified 48K APPLE II Plus. The editor supports most of the commonly used edit commands including find, locate, change, append, insert and delete. Also included are read and write disk commands.

Uppercase letters are entered by preceding them with a CTRL-A. Internally, the program adds 32 to the ASCII value of each lower case letter, thus setting up the string for output to the printer. On the screen, capital letters are converted to the inverse mode while the lowercase letters are converted back to uppercase for display only.

I purchased my APPLE II early in December 1980, and quickly realized that the BASIC language had changed a lot since I had used it last in 1968. The biggest change I noticed was the string handling ability of the new BASIC.

The first application program I decided to write required the use of strings. I quickly found the "write text" and "read text" programs on the master disk and as quickly decided I didn't like them. At work, I make use of text editors on PRIME and UNIVAC computers and find that each of them has certain features which the other doesn't support. So I backed myself into writing a text editor for my APPLE and decided to incorporate the features I liked best from each system.

The program is used the first time to create a text file. The procedure is to hit a carriage return when prompted for "FILE NAME." This puts the program in the input mode. Once the text is entered, a CR puts the program into the EDIT mode. The options available in the EDIT mode are described below. Note that a single letter followed by a space and then any needed parameters is the usual format within the program. In this version, capital letters are typed in by preceding them with a CTRL-A.

The edit options are as follows:

I — Insert new line behind the present line.

C — Change the first sub-string to the second sub-string in this line of text. Sub-strings are separated by /'s. Double //'s can be used to enter a new substring in front of the existing string or to delete the last part of the original string.

A — Append new string to the end of the original string on this line.

P — Print a number of lines. Options include printing all lines from the present position to the end of the file by typing P*.

S — Save file. It is saved with its original name if one has been previously entered. Otherwise, a file name is requested via a prompt. If you give a file name when using S, the new name is used. This is a way of making an image of a text file for backup or modification.

N — If alone the next line is displayed. N +/– NUMB moves the pointer back and forth within file limits.

L — Locate sub-string at any location in any line from the present line to the end of the file.

Q — Quit. Normal program exit.

F — Find sub-string at beginning of any line from the present line to the end of file.

R — Retype present line completely.

H — Help if you have forgotten how to use the program. Can be used at any time.

E — Enter new file name to be edited. Can be used to edit when finished with the first without having to re-run the program.

NN — NN is any valid line number in the file. This is a direct line number access to the entire file.

The program is well REMarked to help any new programmer understand not only what the program does, but also how it does it.

The printer I have is an EPSON MX-80, but I believe this program will work for any printer which supports lower case characters. Until the day this article was written, I had no idea that I could take advantage of the printer's lower case abilities, but my son persisted. This program was modified from my original upper-case only version in about four hours.

One necessary feature of this program is the amount of user error-checking which takes place. As of this writing, I am unaware of any way to make the program bomb. Most of the checks were installed originally, but a few were added when bomb-outs indicated an unexpected pitfall such as typing "DELETE" instead of "D" to delete one line.

If anyone would like to save the effort of typing in the program send me a disk, $3, and an SASE mailer and I will provide a copy of this version and the upper-case only version. My mailing address is:

Joseph N. Wrubel
27 Norwood Lane
Aberdeen, NJ 07747

```
1  REM  *********************
2  REM  *                   *
3  REM  *  LC.EDIT PROGRAM  *
4  REM  *                   *
5  REM  *        BY         *
6  REM  *                   *
7  REM  *   J. N. WRUBEL    *
8  REM  *                   *
9  REM  *                   *
10  REM  *   REV. AUG 1981   *
11  REM  *                   *
12  REM  *********************
13  REM
15  HOME : DIM T$(500)
20  INPUT "FILE NAME :";Z$
24 D$ =  CHR$ (4): REM  CTRL-D
25  REM
26  REM ***********************
27  REM  CR TO ENTER BUILD MODE
28  REM ***********************
29  REM
30  IF  LEN (Z$) = 0 GOTO 1000
32  IF Z$ = "H" THEN 9400: REM LIST INSTRUCTIONS
34  REM
35  REM ***********************
36  REM   LOAD FILE FROM DISK
37  REM ***********************
38  REM
40  PRINT D$;"OPEN";Z$
50  PRINT D$;"READ";Z$
52  REM
53  REM ***********************
54  REM  FIRST ELEMENT FROM DISK IS FILE LENGTH (NUMERIC)
55  REM
57  REM  REMAINDER OF FILE IS IN STRING FORMAT
58  REM ***********************
59  REM
60  INPUT I
70  FOR J = 1 TO I
75  INPUT T$(J): NEXT
80  PRINT D$;"CLOSE";Z$
90 J = 1
100  REM ***********************
101  REM  MAIN REENTRY POINT FROM MOST PROGRAM OPTIONS
102  REM ***********************
103  REM
105  PRINT J;":";
110  GOSUB 250:R$ = T$
112  IF  LEN (T$) = 0 THEN W$ = "": GOTO 121
115 W$ =  CHR$ ( ASC ( LEFT$ (R$,1)) - 32)
116  REM
117  REM ***********************
```

SATURN SYSTEMS 32K RAM BOARD FOR APPLE

Compatible with: Apple II®, Apple II + ®, Microsoft's Z80 Softcard®, DOS 3.2, DOS 3.3, INTEGER Basic®, Applesoft®, PASCAL, FORTRAN, LISA®, Personal Software's VISICALC®

Software included:
1. Relocation of DOS into SATURN 32K board (recovers approximately 10K of main board RAM).
2. Utility package for saving and loading Applesoft® and INTEGER® programs and data on the 32K RAM board; overlaying, chaining.
3. PSEUDO-DISK: Modifies DOS 3.3 to allow use of SATURN 32K RAM board(s) like another disk drive.

COMPREHENSIVE DOCUMENTATION • 1 YEAR WARRANTY

ALL FOR ONLY $239.00

NEW!

MEMORY EXPANSION SOFTWARE FOR VISICALC®

Now you can expand the memory available to Personal Software's 16 sector VISICALC® using the SATURN 32K RAM BOARD!

With VC-EXPAND™ and one or more SATURN 32K RAM BOARDS the memory available to VISICALC® is increased from 18K to:

50K with 1 SATURN 32K BOARD
82K with 2 SATURN 32K BOARDS

In addition, VC-EXPAND™ will utilize your present 16K RAM board to provide 66K of usable VISICALC® memory.

VC-EXPAND supplied on 16 sector disk

ONLY $100.00

DEALER INQUIRIES INVITED
VISA/MasterCard Accepted

ALPHA LOGIC BUSINESS SYSTEMS, INC.
3720 Winston • Hoffman Estates, IL 60195 • (312)870-8230

```
118  REM   CONVERT SINGLE LETTER ENTRY TO NUMERIC
119  REM **********************
120  REM
121  FOR W = 1 TO 13
130  IF W$ =  MID$ ("ICADPSNLQFRHE",W,1) THEN 190
140  NEXT
150  GOTO 1200: REM ENTRY WAS NOT A VALID LETTER
190  ON W GOTO 2000,3000,4000,5000,6000,7000,8000,9000,200,500,2500,9400
     ,750
200  PRINT "PROGRAM COMPLETE": END
210  GOTO 90: REM  CONTINUE RE-ENTRY
250  REM
251  REM **********************
252  REM INPUT STRING SUBROUTINE
253  REM **********************
254  REM
257 T$ = ""
260  GET A$: IF A$ =  CHR$ (13) THEN  PRINT A$;: RETURN
270  IF A$ =  CHR$ (8) AND  LEN (T$) > 1 THEN T$ =  LEFT$ (T$, LEN (T$) -
     1): PRINT A$;: GOTO 260
275  IF A$ =  CHR$ (8) THEN T$ = "": PRINT A$;: GOTO 260
280  IF A$ =  CHR$ (1) THEN  GET A$: INVERSE : PRINT A$;: NORMAL :T$ = T
     $ + A$: GOTO 260
290  IF A$ <  CHR$ (65) OR A$ >  CHR$ (90) THEN T$ = T$ + A$: PRINT A$;:
      GOTO 260
300  PRINT A$;:T$ = T$ +  CHR$ ( ASC (A$) + 32): GOTO 260
350  REM
351  REM **********************
352  REM PRINT A LINE SUBROUTINE
353  REM **********************
354  REM
360  FOR L = 1 TO  LEN (T$):B$ =  MID$ (T$,L,1)
370  IF B$ >  CHR$ (64) AND B$ <  CHR$ (91) THEN  INVERSE : PRINT B$;: NORMAL
     : GOTO 395
380  IF B$ >  CHR$ (96) AND B$ <  CHR$ (123) THEN B$ =  CHR$ ( ASC (B$) -
     32)
390  PRINT B$;
395  NEXT : PRINT : RETURN
450  REM
451  REM ***********************
452  REM STRING DECODE SUBROUTINE
453  REM ***********************
454  REM
460  FOR M = 3 TO  LEN (R$)
470  IF  MID$ (R$,M,1) >  CHR$ (95) AND  MID$ (R$,M,1) <  CHR$ (124) THEN
     R$ =  LEFT$ (R$,M - 1) +  CHR$ ( ASC ( MID$ (R$,M,1)) - 32) +  MID$
     (R$,M + 1)
480  NEXT : RETURN
500  REM ********************
501  REM  FIND STRING ROUTINE
502  REM ********************
503  REM
510  IF  LEN (R$) < 3 THEN 580
520 F$ =  MID$ (R$,3): REM  STRING TO BE FOUND
530  FOR K = J + 1 TO I
540  IF  LEFT$ (T$(K), LEN (F$)) = F$ THEN 570
550  NEXT
560  PRINT "NO FIND": GOTO 90
570 J = K: GOTO 6300
```

```
580  PRINT "YOU MUST ENTER STRING": GOTO 100
750  REM
751  REM ********************
752  REM  ENTER NEW FILE NAME
753  REM ********************
754  REM
755  HOME
760  IF  LEN (R$) < 3 THEN 20
770  GOSUB 450:Z$ =  MID$ (R$,3): GOTO 25
999  REM
1000  REM ******************
1001  REM   BUILD FILE MODE
1002  REM ******************
1003  REM
1005 I = 0:J = 0
1007  PRINT "INPUT"
1010  PRINT J + 1;":";
1020  GOSUB 250:T$(J + 1) = T$
1030  IF  LEN (T$(J + 1)) = 0 GOTO 1100
1040 J = J + 1:I = I + 1
1050  GOTO 1010
1090  REM
1091  REM *****************
1092  REM  ENTER EDIT MODE
1093  REM *****************
1094  REM
1100  PRINT "EDIT": GOTO 100
1200  REM
1201  REM *********************
1202  REM CR TO ENTER INPUT MODE
1203  REM *********************
1204  REM
1205  IF  LEN (R$) = 0 THEN 1500
1206  REM
1207  REM  ********************
1208  REM  VALIDATE LINE POINTER
1209  REM  ********************
1210 W =  VAL (R$)
1215  IF W < 1 OR W > I GOTO 1240
1220 J = W
1230 T$ = T$(J): GOSUB 350: GOTO 100
1240  PRINT "ILLEGAL ENTRY": GOTO 100
1500  REM
1501  REM  ********************
1502  REM       INPUT MODE
1503  REM  ********************
1504  REM
1505  REM  IF AT END OF FILE DO EASY WAY
1507  IF J = I GOTO 1007
1509  REM  THE HARD WAY
1510  PRINT "INPUT"
1515  PRINT J + 1;":";
1520  GOSUB 250
1530  IF  LEN (T$) = 0 GOTO 1100: REM   RETURN TO EDIT MODE
1540  FOR K = I TO J STEP  - 1
1550 T$(K + 1) = T$(K)
1560  NEXT
1570 T$(J + 1) = T$
1580 J = J + 1:I = I + 1
```

```
1590  GOTO 1515
2000  REM
2001  REM *********************
2002  REM   INSERT NEW LINE
2003  REM *********************
2004  REM
2005  IF  LEN (R$) < 3 THEN  PRINT "BAD I": GOTO 100
2010 I = I + 1
2020  FOR K = I - 1 TO J STEP  - 1
2030 T$(K + 1) = T$(K)
2040  NEXT
2050 T$(J + 1) =  MID$ (R$,3)
2060 J = J + 1
2070  GOTO 100
2500  REM
2501  REM *******************
2502  REM     RETYPE LINE
2503  REM *******************
2504  REM
2505  IF  LEN (R$) < 3 THEN  PRINT "BAD R": GOTO 100
2510 T$(J) =  MID$ (R$,3)
2520  GOTO 100
3000  REM
3001  REM *********************
3002  REM  CHANGE PART OF LINE
3003  REM *********************
3004  REM
3005  IF  LEN (R$) < 3 THEN  PRINT "BAD C": GOTO 100
3010 W$ =  MID$ (R$,3)
3020  IF  LEFT$ (W$,1) <  > "/" OR  RIGHT$ (W$,1) <  > "/" THEN 3060
3030  FOR K = 2 TO  LEN (W$) - 1
3040  IF  MID$ (W$,K,1) = "/" GOTO 3070
3050  NEXT
3060  PRINT "MISSING DELIMITERS": GOTO 100
3070 F$ =  MID$ (W$,2,K - 2)
3075 H =  LEN (T$(J))
3080  FOR M = 1 TO H
3090  IF  MID$ (T$(J),M,K - 2) = F$ GOTO 3120
3100  NEXT
3110  PRINT "NO FIND": GOTO 100
3120 G$ =  MID$ (W$,K + 1, LEN (W$) - K - 1)
3125  IF H - M + 1 -  LEN (F$) = 0 GOTO 3160
3127  IF K = 2 GOTO 3170
3128  IF M = 1 GOTO 3190
3130 T$(J) =  LEFT$ (T$(J),M - 1) + G$ +  RIGHT$ (T$(J),H - M + 1 -  LEN
     (F$))
3140  GOTO 6300
3160 T$(J) =  LEFT$ (T$(J),M - 1) + G$: GOTO 3140
3170 T$(J) =  MID$ (W$,3, LEN (W$) - 3) + T$(J): GOTO 3140
3190 T$(J) = G$ +  RIGHT$ (T$(J),H - M + 1 -  LEN (F$)): GOTO 3140
4000  REM
4001  REM **********************
4002  REM APPEND TO PRESENT LINE
4003  REM **********************
4004  REM
4005  IF  LEN (R$) < 3 THEN  PRINT "BAD A": GOTO 100
4010 T$(J) = T$(J) +  MID$ (R$,3)
4020  GOTO 6300
5000  REM
```

```
5001  REM ********************
5002  REM    DELETE LINE(S)
5003  REM ********************
5004  REM
5007 L =  LEN (R$)
5010  IF L > 1 GOTO 5050
5012  REM A "D" ALONE DELETES ONE LINE ONLY
5020  FOR K = J TO I
5030 T$(K) = T$(K + 1): NEXT
5040 I = I - 1:J = J - 1: GOTO 100
5050  IF L = 2 GOTO 5110
5055 N =  VAL ( MID$ (R$,3))
5060  IF N > I - J + 1 THEN 5100
5065  IF N = 0 THEN  PRINT "BAD D": GOTO 100
5070  FOR K = J TO I - N
5080 T$(K) = T$(K + N): NEXT
5090 J = J - 1:I = I - N: GOTO 100
5100  PRINT "DELETE TOO BIG": GOTO 100
5110  PRINT "ILLEGAL DELETE": GOTO 100
6000  REM
6001  REM *******************
6002  REM  PRINT SOME LINES
6003  REM *******************
6004  REM
6007  IF  LEN (R$) < 2 THEN 6300
6010 NUM$ =  MID$ (R$,2)
6020  IF NUM$ = "*" GOTO 6150
6030 NUM =  VAL (NUM$)
6035  IF NUM = 0 THEN T$ = T$(J): GOSUB 350: GOTO 100
6040  FOR K = J TO J + NUM - 1
6050 T$ = T$(J): GOSUB 350:J = J + 1
6060  IF J > I GOTO 6100
6070  NEXT
6075 J = J - 1
6080  GOTO 100
6100  PRINT "EOF:";I;" LINES"
6104  REM
6105  REM  THE END OF FILE WAS FOUND
6106  REM
6110  GOTO 90
6150  REM
6151  REM ********************
6152  REM  IS PRINTOUT WANTED
6153  REM ********************
6154  REM
6160  PRINT : INPUT "PRINTOUT?";PR$
6170  IF  LEFT$ (PR$,1) = "Y" THEN 6350
6180  IF  LEFT$ (PR$,1) = "N" THEN 6200
6190  PRINT : PRINT "TRY AGAIN": GOTO 6160
6200  FOR K = J TO I
6210 T$ = T$(K): GOSUB 350: NEXT
6220  GOTO 6100
6300 T$ = T$(J): GOSUB 350: GOTO 100
6350  REM
6351  REM *********************
6352  REM   PRINT ENTIRE FILE
6353  REM *********************
6354  REM
6360  PRINT D$;"PR#1": PRINT  CHR$ (9);"80N"
```

```
6370  FOR K = J TO I
6374  REM  IF PERIOD SKIP A LINE
6375  IF T$(K) = "." THEN  PRINT : GOTO 6385
6378  IF  LEFT$ (T$(K),4) = ".    " THEN T$(K) = " " +  MID$ (T$(K),2)
6380  PRINT T$(K)
6385  NEXT
6390  PRINT D$;"PR#0": GOTO 6100
7000  REM
7001  REM ******************
7002  REM      SAVE FILE
7003  REM ******************
7004  REM
7006  IF  LEN (R$) > 2 THEN  GOSUB 450:Z$ =  MID$ (R$,3)
7008  IF  LEN (Z$) <  > 0 THEN 7015
7010  PRINT : INPUT "FILE NAME ?";Z$
7012  IF  LEN (Z$) = 0 THEN 7010
7015  PRINT D$;"OPEN";Z$
7020  PRINT D$;"DELETE";Z$
7030  PRINT D$;"OPEN";Z$
7040  PRINT D$;"WRITE";Z$
7050  PRINT I
7060  FOR J = 1 TO I
7070  PRINT T$(J): NEXT
7080  PRINT D$;"CLOSE";Z$
7090  GOTO 90
8000  REM
8001  REM *********************
8002  REM   RELATIVE MOVEMENT OF POINTER
8003  REM *********************
8004  REM
8005  IF R$ <  >  CHR$ (110) THEN 8030: REM   A TRANSLATED "N"
8010 J = J + 1
8015  IF J > I THEN  PRINT "EOF:";I;" LINES": GOTO 90
8020 T$ = T$(J): GOSUB 350: GOTO 100
8030 V =  VAL ( MID$ (R$,2))
8040  IF V + J > I OR V + J < 1 GOTO 8100
8050 J = J + V
8060 T$ = T$(J): GOSUB 350: GOTO 100
8100  PRINT "MOVE TOO BIG": GOTO 100
9000  REM
9001  REM **********************
9002  REM   LOCATE STRING
9003  REM **********************
9004  REM
9007  IF  LEN (R$) < 3 THEN  PRINT "BAD L": GOTO 100
9010 F$ =  MID$ (R$,3)
9020  FOR K = J + 1 TO I
9030  FOR M = 1 TO  LEN (T$(K)) -  LEN (F$) + 1
9040  IF F$ =  MID$ (T$(K),M, LEN (F$)) GOTO 9070
9050  NEXT M: NEXT K
9060  PRINT "NO FIND": GOTO 90
9070 J = K: GOTO 6300
9400  REM
9401  REM **********************
9402  REM        HELP USER
9403  REM **********************
9404  REM
9405  HOME
9407  PRINT : PRINT  SPC( 9);"TEXT EDITING PROGRAM"
```

```
9410  PRINT : PRINT "EACH SINGLE CHARACTER INSTRUCTION SHOWN"
9415  PRINT "BELOW IS TO BE FOLLOWED BY A SPACE AND"
9420  PRINT "AND THEN ANY NEEDED PARAMETERS."
9425  PRINT : PRINT "TO START A NEW FILE, PUSH RETURN WHEN"
9430  PRINT "YOU ARE PROMPTED FOR THE FILE NAME."
9435  PRINT "YOU MAY THEN ENTER YOUR TEXT FILE LINE
9440  PRINT "BY LINE.  WHEN DONE, PUSH RETURN AGAIN"
9445  PRINT "TO ENTER THE EDIT MODE.
9450  PRINT : PRINT  SPC( 4);"** PUSH ANY KEY TO CONTINUE **"
9460  GET G$
9505  HOME
9510  VTAB 2: HTAB 10
9515  PRINT "TEXT EDITING PROGRAM"
9520  PRINT : PRINT "CODE      FUNCTION"
9525  PRINT : PRINT " I        INSERT NEW LINE OF TEXT"
9527  PRINT " ","BEHIND THE PRESENT LINE"
9530  PRINT : PRINT " C        CHANGE THE FIRST STRING TO "
9535  PRINT  SPC( 9),"THE SECOND, USE /'S TO"
9540  PRINT " ","SEPARATE STRINGS"
9545  PRINT : PRINT " A"; SPC( 7);"APPEND STRING TO END OF LINE"
9550  PRINT " ","LEAVE 1 SPACE BETWEEN"
9555  PRINT " ","THE A AND THE STRING"
9560  PRINT : PRINT " D"; SPC( 7);"DELETE 'N' LINES, IF N OMITTED,";
9565  PRINT " ","JUST THIS LINE IS DONE"
9570  PRINT : PRINT " P"; SPC( 7);"PRINT 'N' LINES FROM HERE"
9575  PRINT " ","USE P* TO LIST ALL"
9580  PRINT : PRINT  SPC( 6);"** PUSH ANY KEY TO CONTINUE **"
9585  GET G$
9590  HOME : PRINT : PRINT "CODE     FUNCTION"
9595  PRINT : PRINT " S"; SPC( 7);"SAVE FILE WITH NAME ENTERED"
9600  PRINT " ","IF NO NAME IS ENTERED"
9605  PRINT " ","USE ORIGINAL FILE NAME"
9610  PRINT : PRINT " N"; SPC( 7);"NEXT LINE +/- NUMB IS PRINTED"
9615  PRINT : PRINT " L"; SPC( 7);"LOCATE STRING FROM HERE"
9620  PRINT " ","TO END OF FILE"
9625  PRINT : PRINT " Q"; SPC( 7);"QUIT"
9630  PRINT : PRINT " F"; SPC( 7);"FIND STRING AT START OF ANY"
9635  PRINT " ","LINE FROM HERE TO END"
9640  PRINT : PRINT " R"; SPC( 7);"RETYPE PRESENT LINE"
9645  PRINT : PRINT " H"; SPC( 7);"HELP PROVIDED VIA THIS LIST"
9650  PRINT : PRINT  SPC( 7);"** PUSH ANY KEY TO CONTINUE **"
9651  GET G$
9653  HOME : PRINT : PRINT "CODE     FUNCTION"
9655  PRINT : PRINT " E"; SPC( 7);"NAME FILE TO BE EDITED"
9660  PRINT : PRINT "(CR)"; SPC( 5);"USE CARRIAGE RETURN TO
9665  PRINT  SPC( 9);"ENTER INPUT MODE"
9690  PRINT : PRINT : HTAB 5: PRINT "** PUSH ANY KEY TO CONTINUE **"
9695  GET G$: GOTO 100
```

©

COMPUTE! OVERVIEW:

Individual Tax Plan

The "Individual Tax Plan" program by Aardvark Software, Incorporated is a highly sophisticated piece of computer software for the Apple computer system (II or Plus) with at least 48K of RAM and two disk drives, DOS 3.3 or PASCAL. It also nicely lends itself to the computerist who, in essence, does not have a working knowledge of computers. As long as the manual is at least previewed, one will not have any trouble running this program.

It is a well-designed, easy to use system for comparing different filing alternatives in order to minimize the income tax liability for an individual taxpayer. It does an effective job of allowing a comparison of numerous different tax preparation schemes at one time. It does not, however, do all of the work and calculations necessary to complete a tax return. Perhaps a better name for the software package would have been "Individual Tax Comparison Scheme."

Up to five alternative tax preparation schemes may be entered at once. One alternative, for example, could include income averaging with schedule G while others could compare filing jointly vs. filing singly for a married couple. Side-by-side comparison of the calculated taxes for each of the alternatives is effectively done by the program. The program is only of value, however, after an individual has calculated many of the numbers that belong on the tax return. For example, tax credits is a single item to be entered. The taxpayer (or tax return preparer) must determine the tax credits for child care expenses and energy-saving expenses (each a percentage of actual expenses and each subject to dollar limitations and other limiting factors), and add them together. This sum is the value that is entered into the "Individual Tax Plan."

It should be stressed that this program is not oriented towards layman use, but towards the tax professional, who has had previous tax preparation exposure. To effectively use this powerful tool one must have a working knowledge of possible tax alternatives to pursue.

Updates

Should changes in federal tax law occur in a calendar year, Aardvark Software will make available revised programs reflecting these changes. Revisions will cost $50.00 and can be obtained from local Aardvark Software dealers. Annual updates reflecting changes in tax law and including program enhancements will be made available on or before November 1st of each calendar year.

Back-up copies of the included program and data disks are allowed using the standard Apple copy program. You should be able to save between 50 and 75 Tax Plan cases on each copy of the data diskette.

Using The Program

During operation of the program the user enters data for up to 74 categories, such as filing status, interest, charitable contributions, and "long term capital gains-post 6/8/81." Unfortunately, the documentation does not follow the program exactly in the identification of the different categories. Items 12 through 32 are misidentified, most of the numbers being off by one. Once the changes are marked on two of the four pages which identify the various categories, there is no difficulty finding the various items, but the problem should never have occurred.

For each category a value can be independently entered for each alternative, or programming options can be used to calculate values for different alternatives. For example, if $10,000 is entered for the first filing alternative, then the remaining alternatives are calculated by the program at 20% increments by simply entering "P20" for percent-20. Other options include "X" if only the next alternative is to be calculated on a percentage basis or "I" for "increment" if all subsequent alternatives are to differ from each other by a specific dollar amount.

After all of the data is entered, the program takes a few seconds to calculate the taxes for all of the alternatives. Any two alternatives (in any order) may be printed as hard copy for easy comparison of the alternatives in different columns. In a strange departure from the easy to use options, here "999" must be entered to indicate that the numbers of all of the desired options have selected. RETURN would have been far easier to use.

Flexibility

One of the strengths of this software package is the ease with which a user can move from one part of the program to another. From a main menu single digit numbers are used to reach further menus which identify specific activities. Several options are offered for moving from category to category for data entry. To "select" a specific category "S" can be typed followed by the number of the category. To move "forward" to the next category "F" is used and "B" is used to "back up." For many of the categories up to ten numbers can be entered — five for the taxpayer and five for the spouse. The program is smart enough to fill up all of the alternatives with the value given for the first alternative unless it is specifically given new values for subsequent alternatives. To move from one specific alternative to another "U" is used to go "up" and "D" is used to go "down." No control keys are required here — the editing is very easy to use.

ESCape can be used at nearly any time to exit from data entry and save on disk all of the values that have been entered for all of the alternative schemes. One minor irritant here is that the Pascal volume numbers are used to specify the disk drives. The documentation explains that disk drive #1 needs to be specified as volume four and so on, but the program should have been written to accept simple drive numbers. The name that is given for the file is first checked against those currently on the disk in order to prevent inadvertent over writing of a file that should be maintained. An option is also provided to see the directory of items that have been stored.

There are no charts included to indicate which of the 74 possible tax input questions are to be entered if, for example one were filing "married with a joint return." A glossary of terms would also be a welcome addition. However, execution speed is an outstanding feature of this program. All calculations are performed in under 60 seconds, regardless of complexity. The program appears to be written entirely in machine code, which would account for its exceptional speed.

While the ranges of input data appear to be sufficiently checked, disk error codes are vaguely defined. If RESET is pressed, all existing data not saved on disk is lost and the program requires rebooting for continued operation. This can be most annoying and could possibly prove fatal if done during a disk storage operation.

The Documentation

Documentation for the individual tax plan program consists of an attractive 3-ring binder with a 31 page illustrated instruction manual which includes a simple appendix and printouts. The documentation, although sufficient for the tax professional, is not designed to be a comprehensive overview of tax preparation for the layman.

With the exception of the misnumbered categories, the documentation is clear and complete. About ten pages are used to lead the user through

ECHO][™ *SPEECH SYNTHESIZER*

UNLIMITED VOCABULARY AT YOUR FINGERTIPS

Give your Apple* something to talk about with an ECHO][Speech Synthesizer. The ECHO][offers intelligible voice-output while using a minimum of RAM. The ECHO][uses LPC technology pioneered by Texas Instruments, coupled with a phoneme-based operating system allowing you to create any vocabulary desired. Variable stress, pitch and volume let your Apple ask questions or make exclamations while also allowing for optimal quality. Speech can easily be added to BASIC programs with PRINT statements.

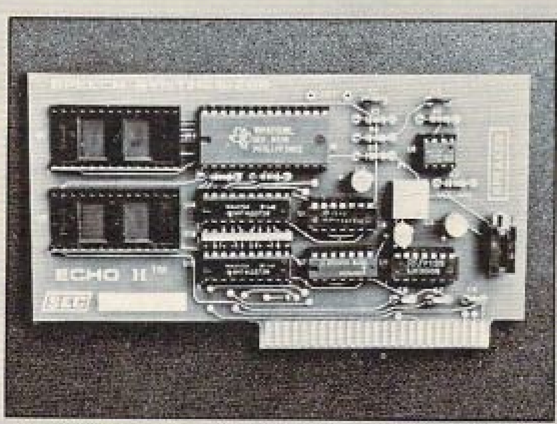
The applications of synthesized speech are limited only by your imagination. From educational programs to games to business applications, speech-output adds a new dimension to your Apple's capabilities.

The ECHO][Speech Synthesizer comes complete with speaker and cable, instruction manual, speech editing features and a sample vocabulary. The ECHO][requires 48K, Applesoft and at least one disk drive. Suggested list price is $225.

For further information, contact your dealer or Street Electronics Corporation.

Dealer inquiries welcome.

* Trademark of Apple Computer Company



SEC STREET ELECTRONICS CORPORATION

3152 E. La Palma Ave., Suite D
Anaheim, CA 92806 (714) 632-9950

two simple examples that do a good job of demonstrating how to move about in the program. Sample printed output for each of these examples is given in an Appendix (misidentified as Appendix "B"). About ten more pages are used to specifically describe the program options and to identify the various categories for data entry. Throughout the instructions, 27 photographs of screen images appear. The photographs were apparently taken with a wide angle lens and therefore appear distorted, but they are readable and provide an accurate representation of what the program displays.

General Overview

Panelist #1: "Negative and detracting hindrances:

(1) There should be a subroutine within the program which would enable the user to enter directly into a mini-directory to review a directive or procedure.

(2) The ability to only do the filing status routine should be looped so that only an individual taxpayer entry is verified and utilized when there is no spouse involved.

(3) Provision to exercise the use of only one disk drive should be available when only one is involved or necessary.

(4) An ending directive within the program (other than in the manual) should be provided after all statistics have been entered.

(5) A 'short form' alternative option could be incorporated.

Positive and useful aspects:

(1) Exceedingly fast access time.

(2) Ease of use in the main menu parameters.

(3) Printer parameters and linefeed status changes.

(4) Aardvark's updating procedures)annually or when the tax structure/laws change)

(5) Comparative analysis of defined numerical statistics to take advantage of the lowest tax amount to be paid.

(6) The 'step' feature: accessing forward and backward through the program via a single keystroke.

(7) Ability to access any part of the program by entering the input of the area and return.

(8) User defined changes: save data (Y/N), screen or printer display, program user return (ability to re-enter your numerical statistics and make any changes necessary in any of the alternative figures prior to executing the calculations).

(9) Ability to handle positive and negative integers as well as figuring out its compound percent.

(10) User ability to make any and all necessary backup copies in the event of catastrophes."

Panelist #2: "The software is easy to use and effectively compares calculations done on the basis of different tax preparation schemes. It does not do all of the calculations that a taxpayer needs to do, nor does it identify a correspondence between specific line numbers on form 1040 and the categories within the program. The software package could be very useful for professional tax preparers, but is not likely to be worth the expense for an ordinary taxpayer. For someone with substantial capital gains to declare, it could be helpful, but that person is probably going to benefit from advice from a professional anyway. Whom should you select as that professional? Someone who has an Apple and Aardvark's Individual Tax Plan."

Panelist #3: "This program was designed by a group of CPA's with over 17 years of "Big Eight" experience to meet the needs of the professional tax practitioner.

This program is not, nor was it designed to be, everyone's answer to H & R Block. With some additional documentation, a much wider range of people could benefit from it. While not intended for the layman, the professional tax preparer should find this program an outstanding value."

Sample Output

Table 1.

```
1981                            ALTERNATIVE
                                     1
FILING STATUS                      JOINT
EXEMPTIONS                             2
WAGES, SALARIES                   28,480
INTEREST AFTER EXCLUSION             350
DIVIDENDS AFTER EXCLUSION              0
CAPITAL GAIN/LOSS                      0
PARTNERSHIP INCOME/LOSS                0
OTHER INCOME/LOSS                  2,000
                                --------
TOTAL INCOME                      30,830
ADJUSTMENTS TO INCOME              1,600
                                --------
ADJUSTED GROSS INCOME             29,230

DEDUCTIONS
MEDICAL & DENTAL EXPENSES            170
STATE & LOCAL INC TAXES            1,681
OTHER TAXES                            0
INTEREST EXPENSE                   1,690
CHARITABLE CONTRIBUTIONS             943
CASUALTY LOSS                      1,090
MISCELLANEOUS                        787
                                --------
TOTAL DEDUCTIONS                   6,361
ZERO BRACKET AMOUNT                3,400
                                --------
EXCESS ITEM. DEDUCTIONS            2,961
```

```
TAX TABLE INCOME                          26,269
EXEMPTIONS TIMES $1,000                   ** N/A
                                   -------------
TAXABLE INCOME                            ** N/A
                                   -------------
TAX - TAX TABLES/XYZ                       4,359
TAX - QUAL. CAP. GAINS                    ** N/A
TAX - INCOME AVERAGING                         *
TAX - MAXIMUM TAX                         ** N/A

TAX SELECTED                               4,359
ADDITIONAL TAXES                               0
                                   -------------
GROSS REGULAR TAX                          4,359
CREDITS                                        0
                                   -------------
NET REGULAR TAX                            4,359
MINIMUM TAX                                    0
ALTERNATIVE MINIMUM TAX                        0
OTHER TAXES                                    0
                                   -------------
TOTAL TAX LIABILITY                        4,359
FEDERAL PAYMENTS                           4,998
                                   -------------
BALANCE DUE (REFUND)                        -639
```

Individual Tax Plan. Aardvark Software, Inc., 783 North Water Street, Milwaukee, WI 53202. 48K Apple, two disk drives, DOS 3.3 or Pascal, $250. ©

Common Cents

FORECASTER

With your computer, you can forecast the stock market like the Wall Street experts. FORECASTER, a mutual and money market fund tracker, monitors the stock market and helps you make timely decisions to achieve a high and safe growth of your investment dollars. Our program tracks, estimates and predicts future movements in the stock market and selected mutual funds based on an extensive data base.

- ● AVOID CONFUSION
- ● ACHIEVE HIGH YIELDS
- ● PRESERVE CAPITAL
- ● LOW RISK
- ● TOP INTEREST RATES
- ● TAKES LESS THAN ½HOUR PER WEEK

FORECASTER is written in simple, clear language. No advanced knowledge of mathematics, computer programming, or investing experience is required. Send $10.00* for programming report and user's manual, or $75.00* for complete software package. Please specify Apple II+ (Applesoft), CP/M Basic 80, or TRS 80 Level II. (Apple II+ and Applesoft are trademarks of Apple Computers, CP/M is a trademark of Digital Research, Basic 80 is a trademark of MicroSoft, and TRS 80 is a trademark of Tandy Corp.

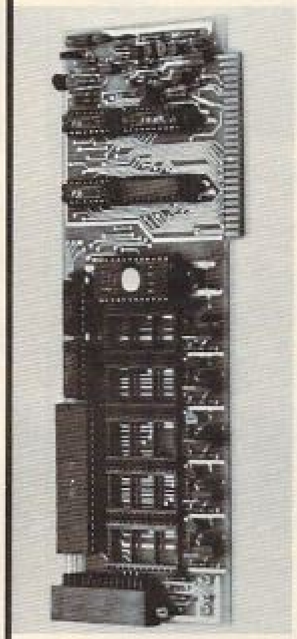
* Add $2.00 for postage and handling
New Mexico residents also add 4% tax

META SOFTWARE ENGINEERING
P.O. Box 18056
Albuquerque, NM 87115
D 1

OMNI

The "Everything" I/O Board for the Apple II & II+

OMNI is a multi-function input/output board for the Apple II or II+ computer. It provides, on a single board, most of the "missing" features needed to make the Apple a complete computer. With OMNI your Apple can have:



- Parallel I/O with handshaking
- RS232 Level serial I/O (software driven)
- 24 Hr. Real Time Clock with Alarm
- 2K EPROM with graphics, I/O driver, and screen editor firmware
- 256 Byte PROM supervisory firmware
- Six 2K PROM/EPROM expansion sockets software selectable (one socket used for Eclectic firmware, 5 available for user)

INPUT:
- Generate full ASCII character set from keyboard
- Optional shift key detection
- User-definable "soft" keys with screen legends
- Integrated text line editor full cursor movement, insertion/deletion modes
- Key legend stickers included
- Demonstration Diskette with programming examples and a Soft Character Editor

OUTPUT:
- Full 96 character ASCII display
- Concurrent 64 user-definable "soft" characters
- Optional character overstrike and EOR on background
- Optional double-width color characters
- Character rotation in 90° steps

Never before have so many functions been available on a single board.

OMNI was designed with one major goal in mind, flexibility. The OMNI system consists of some extremely simple but very sophisticated hardware, a large amount of powerful firmware (programs permanently residing in Read Only Memory chips), and an equally extensive amount of software (programs residing on diskette that are loaded into RAM as needed). In addition, OMNI comes with extensive documentation.

ECLECTIC SYSTEMS CORPORATION

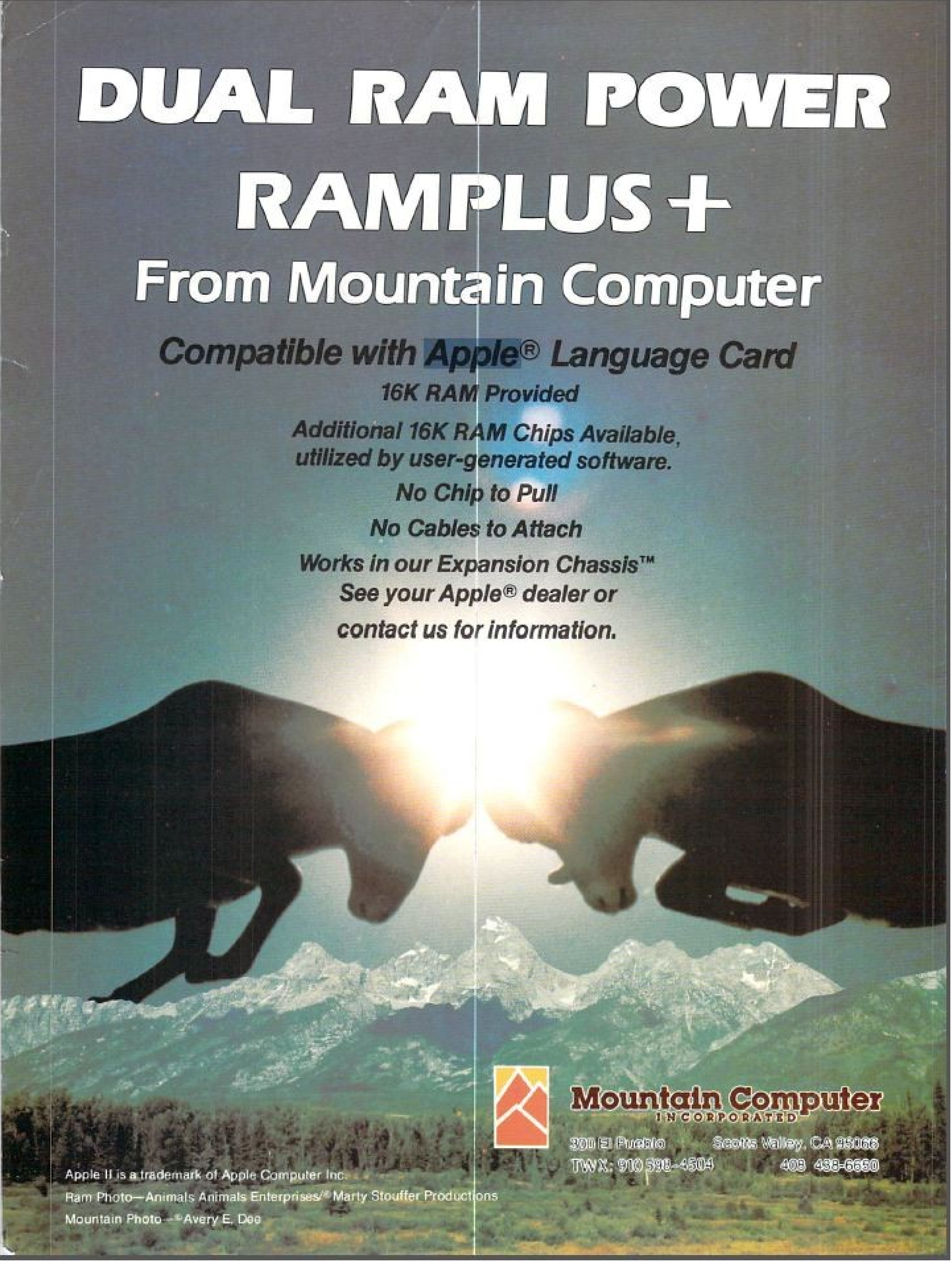
To order TOLL FREE: 1-800-527-3135
or Order by Mail Below

Eclectic Systems Corporation, P.O. Box 1166, Addison, TX 75001

Here's my order for OMNI at $268 plus $3 for shipping and handling (UPS surface, unless specified), $5 overseas.

☐ My check is enclosed.
☐ Please charge my VISA ____ Mastercard ____
Account # ________ Expires ________

Name ____________________
Address ____________________
City/State ____________ Zip ________
Signature ____________________



DUAL RAM POWER RAMPLUS+

From Mountain Computer

Compatible with Apple® Language Card

16K RAM Provided

Additional 16K RAM Chips Available, utilized by user-generated software.

No Chip to Pull

No Cables to Attach

Works in our Expansion Chassis™

See your Apple® dealer or contact us for information.

Mountain Computer
INCORPORATED

300 El Pueblo Scotts Valley, CA 95066
TWX: 910 598-4504 408 438-6650

Apple II is a trademark of Apple Computer Inc.

Ram Photo—Animals Animals Enterprises/© Marty Stouffer Productions

Mountain Photo—© Avery E. Dee

TASC.™ The Applesoft* Compiler. It turns your Apple into a power tool.

Step up to speed. TASC, the Applesoft Compiler, converts a standard Applesoft BASIC program into super-fast machine code. By increasing program execution speed up to 20 times, Microsoft gives you a power tool for Applesoft BASIC programming.

Highest capacity available. TASC will compile and run larger programs than any other Applesoft Compiler. As a disk-based system, it doesn't require the simultaneous presence of compiler and program in memory. The memory you save allows you to compile significantly bigger programs.

Power without bulk. Code expansion of up to 100% severely restricts other compilers. TASC's special code compression schemes typically limit code expansion to only 25%. You'll really appreciate that with complex programs or programs that utilize Apple's hi-res graphic pages.

More BASIC power. TASC's powerful new commands increase Applesoft BASIC programming capability. Chain with COMMON allows compiled programs to share variables, so a main menu supports several programs in a single runtime environment.

TASC's True Integer Arithmetic and Integer FOR...NEXT capabilities maximize the execution speed of compiled programs. TASC's near total compatibility with Applesoft speeds compilation of existing programs with little or no modification.

What about mistakes? You perfect your programs interactively with Applesoft. If something does slip by, TASC recovers from errors discovered in compilation and traps all runtime errors. It even permits graceful interruptions during compilation.

See for yourself. Ask for a demonstration of TASC at your Microsoft dealer. Discover the software package that turns your Apple into a power tool.

MICROSOFT
CONSUMER PRODUCTS

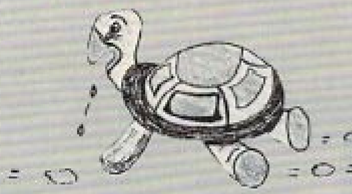
A Division of Microsoft Inc.
10700 Northup Way • Bellevue, WA 98004

*Applesoft is a trademark of Apple Computer, Inc.

Mock Turtle

is crying. Why? Because he has learned that a leading microcomputer manufacturer is planning to market an imitation of **M.I.T.'s Logo** for Apple II*. He thinks that Alice and the world's children deserve the genuine article. Namely,

M.I.T. Logo for Apple II*



Krell's **M.I.T. Logo** for Apple II* is a copyrighted product of the Massachusetts Institute of Technology. Logo was developed under a grant furnished by the National Science Foundation. Krell's **M.I.T. Logo** for the Apple* includes many special features such as Beaver Graphics and Krell Instant Logo Tutor Package. The entire package is fully documented for teachers and students. Requires 64K **$179.95**

RELATED BOOKS

Apple Logo by Harold Abelson (McGraw Hill): indispensable	**14.95**
Mindstorms by Seymour Papert (Basic Books)	**12.95**
Computer Connections by Jean N. Nazzaro, Editor (ERIC)	**16.65**

Memory expansion boards to 64K for Apple II*. **130.00**

FREE BONUS with purchase of $300.00 or more:
Applesoft Tutor Series

PROGRAMS AVAILABLE FOR TRS-80, APPLE II, PET & ATARI

N.Y.S. residents add sales tax.

* Trademark of Apple Corp.

KRELL SOFTWARE CORP.

"The State of the Art in Educational Computing"

21 Millbrook Drive • Stony Brook, New York 11790

(516) 751-5139

Plotting Polar Graphs With The Apple II

Marvin L. De Jong
The School of the Ozarks
Pt. Lookout, MO 65726

You do not need long programs to make a computer perform a useful task in teaching mathematics. One of the more arduous tasks in trigonometry or analytic geometry is graphing functions in polar coordinates. For many polar curves, this task takes a lot of time, and not much learning takes place. On the other hand, it is an ideal task for the computer, the program to plot a polar graph is easily understood by the students, and it gives them a tool with which they can experiment with many graphs. Program 1 shows the simplest possible version. We shall discuss it shortly, but first here is a brief explanation of what we are trying to accomplish.

Suppose we have a relation between R, the distance from a point called the *pole*, and θ (Greek symbol theta), the angle measured counterclockwise from the *polar axis*. The pole is analogous to the *origin* in X-Y Cartesian coordinates, and the polar axis lies along the *X-axis*. The relation between R and θ is usually described by an equation of the form

R = F(θ).

The equation

R = 90*SIN(2*θ)

is one example. Refer to Figure 1 for an illustration of some of these concepts, including a graph of the equation R=90*SIN(2*θ), called a *four-leaved rose*.

The key to using a computer to graph polar coordinates is the transformation formulas

X = R*COS(θ)
Y = R*SIN(θ)

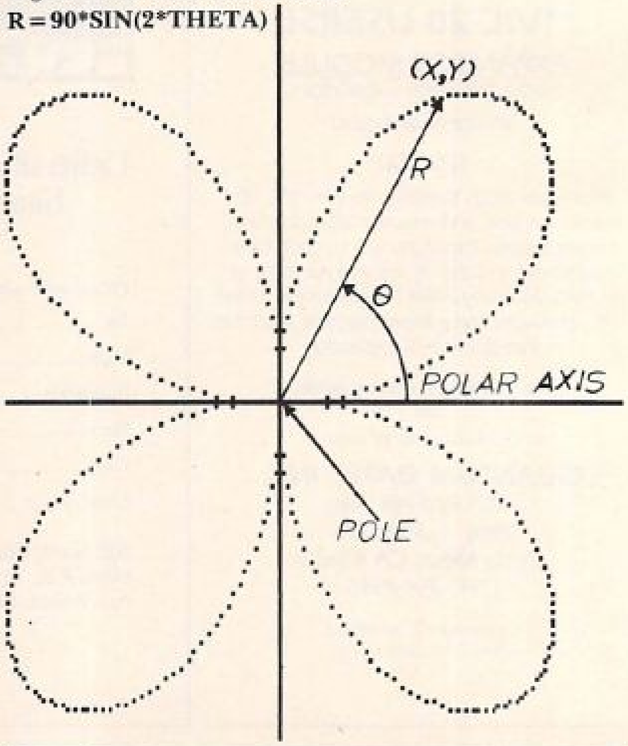
and, of course, the computer's ability to perform a PLOT X,Y instruction.

A "bare bones" approach to plotting polar graphs is given in Program 1. The student inputs the starting angle and the angle at which the graph is to end. These angles are in degrees. Line 30 initializes the HIRES mode with text on the lower part of the screen of the video monitor. Line 60 converts the angle to radians (pi radians = 180°).

Line 70 in Program 1 is the equation to be graphed. The entire program may be left unchanged while line 70 is modified to graph a large variety of polar functions.

Line 90 and 100 convert the polar coordinates (R,θ) to X-Y coordinates. Note that since the origin of the Apple II coordinate system is in the upper left-hand corner of the screen, we have *translated* it so the origin of our coordinate system is at (85,85). Furthermore, since Y is positive *downward* on the Apple, and we would prefer the more traditional "Y positive upward" convention, we use a negative sign in the Y-transformation equation. The results are plotted with the instruction on line 120. The instruction on line 130 increments the angle by one degree. Points will be continued to be plotted until

Figure 1. A Four-Leaved Rose.
R = 90*SIN(2*THETA)



VISICALC® MEMORY EXPANSION SYSTEM FOR THE APPLE II

Now you can expand the memory available to Personal Software's 16 sector VISICALC® using **the SATURN 32K RAM BOARD and VC-EXPAND™.**

With VC-EXPAND™ and one or more **SATURN 32K RAM BOARDS,** the memory available to VISICALC® is increased from 18K to:

- **50K with 1 SATURN 32K BOARD**
- **66K with 1 SATURN 32K BOARD and your present 16K RAM card.**
- **Even more memory available with the SATURN 128K BOARD**

Software is included to relocate DOS onto the 32K RAM BOARD and to allow its use as a fast disk drive. Compatible with existing software in place of a 16K RAM card.

Apple II is a registered trademark of Apple Computer, Inc.
Visicalc is a registered trademark of Personal Software, Inc.
Copyright 1981 Alpha Logic Business Systems, Inc.

COMPREHENSIVE DOCUMENTATION
1 YEAR WARRANTY

Total Price $339.00
Also available separately
Saturn 32K RAM BOARD $239
VC-EXPAND™ $100
Saturn 128K RAM BOARD $599

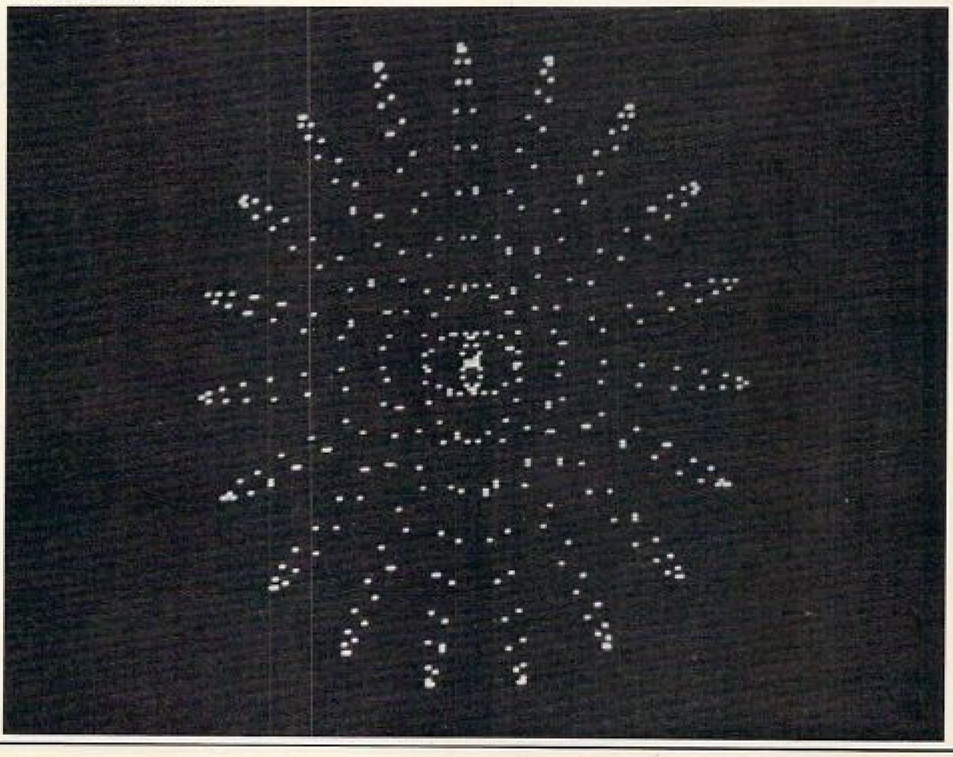
Dealer inquiries invited
Visa and MasterCard accepted

ALPHA LOGIC
BUSINESS SYSTEMS, INC.
3720 Winston Drive
Hoffman Estates, IL 60195
(312) 870-8230

the angle exceeds the ending angle. The student can watch the points being plotted and see the corresponding R and θ values printed underneath the graph.

A photograph of the screen of the video monitor after the graph R = 85*SIN(19*THETA) was

Figure 2: A Nineteen-Leaved Rose



plotted is shown in Figure 2. Notice that different X and Y scale factors on the screen produce a slight distortion that is not important as far as the present application is concerned.

Of course, it is always possible to add a few bells and whistles. Program 2 represents a few non-essential, but nice, additions to the first program. The coordinate axes are drawn and the X and Y values are rounded to their nearest integer values before plotting. Also, we have made use of the entire screen with the HGR2 instruction on line 30. The scale of the graph was reduced so that we could plot the finished result on our little printer. If you are using a video monitor or a large printer, then you will want to keep the scale as large as possible (replace all the 80's with 90's).

Some of our results are given in the figures that follow. In Figure 3 we show a graph of R=80*SIN(3*THETA) a *three-leaved rose.* Figure 4 is a graph of a *13-leaved rose,* R=80*SIN(13*THETA). The *cardioid* R=40*(1+COS(THETA)) is illustrated in Figure 5. Figure 6 is the famous *Spiral of Archimedes,* R=6*THETA. Figure 7 is similar, but not identical to the *Limacon of Pascal.* We chose R=80*COS(THETA/3) for this figure. Figure 8 illustrates the *Litus* described by the equation R=25*(2+SIN(3*THETA)). Figure 9 has no name, but its equation is R=25*(2+SIN(3*THETA)).

Finding where two polar curves intersect is sometimes difficult. If you have a printer you can simply graph the polar curves, overlay their graphs, and find approximate points of intersection.

Students seem to enjoy working with these programs. They are simple enough so the students can modify the various parameters rather easily, giving them a chance to experiment freely. At the

Figure 3. A Three-Leaved Rose.

R=80*SIN(3*THETA)

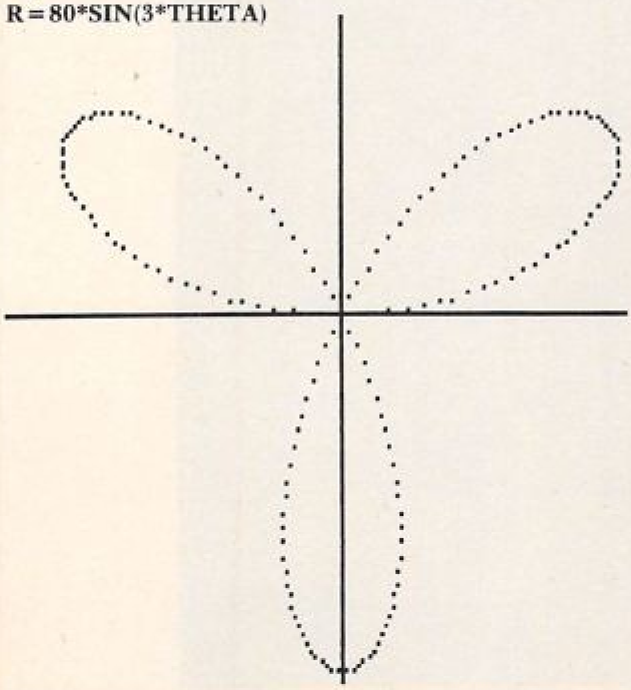


Figure 4. A Graph of R=80*SIN(13*THETA), A 13-Leaved Rose.

R=80*SIN(13*THETA)

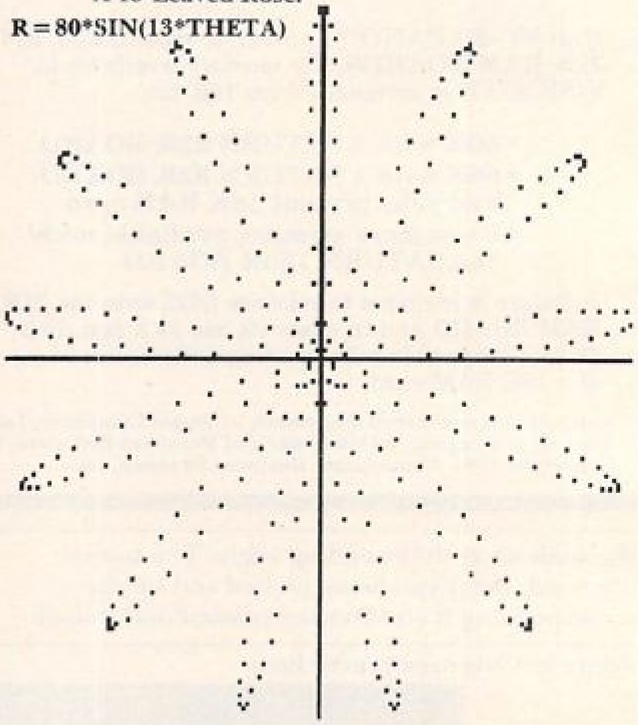
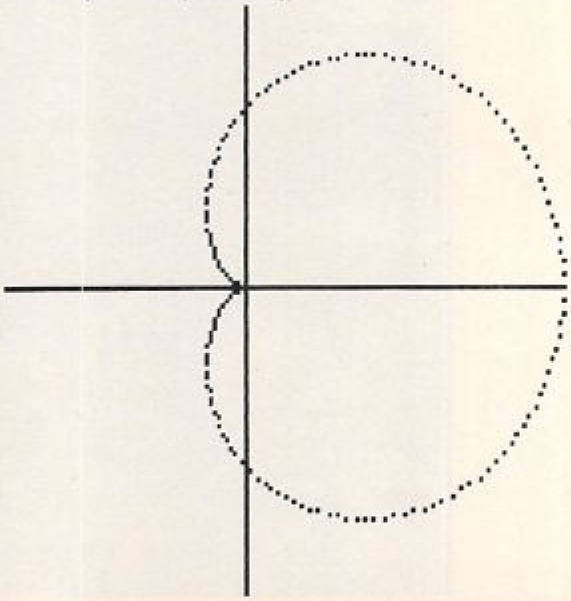


Figure 5. The Cardioid R=40*(1+COS(THETA)).

R=40*(1+COS(THETA))



very least, the programs release them from the drudgery of plotting points by hand.

Figure 6. Spiral of Archimedes with R=6*THETA.

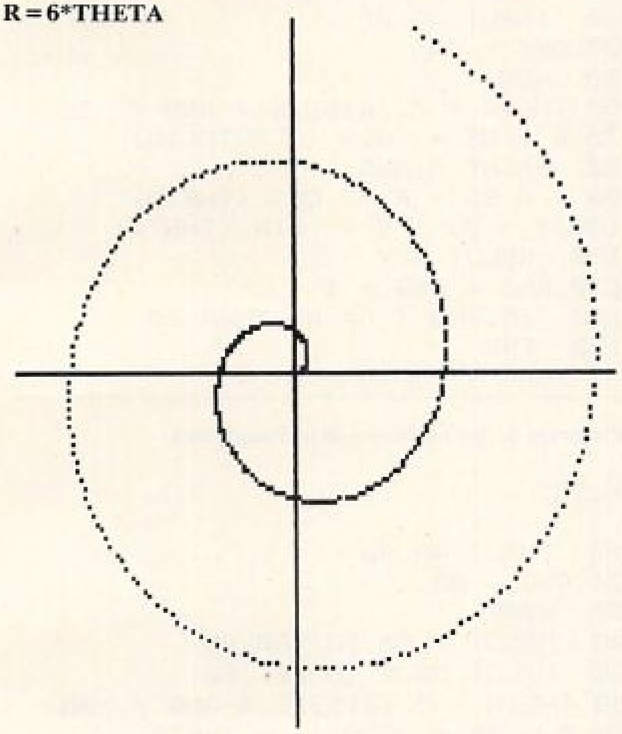
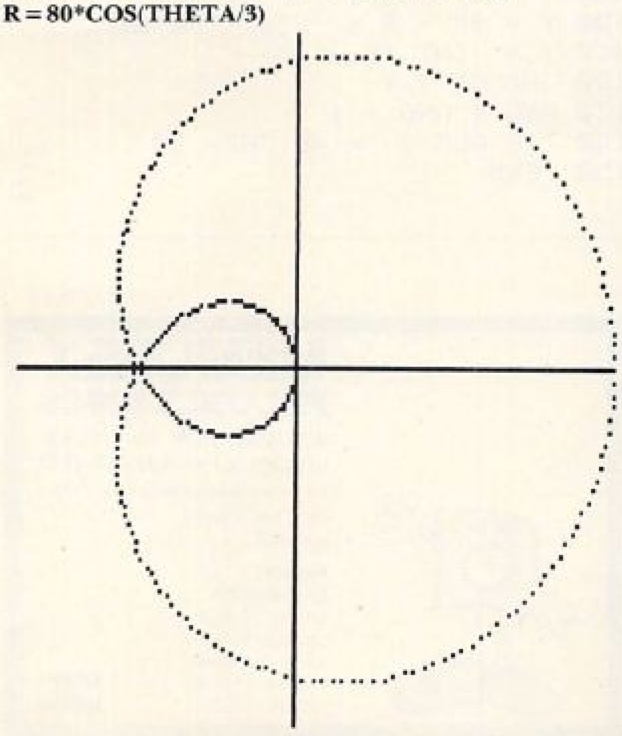


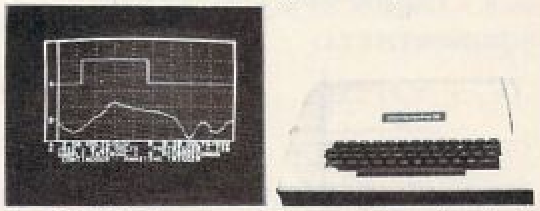
Figure 7. A Graph of R=80*COS(THETA/3).



APPLESCOPE

DIGITAL STORAGE OSCILLOSCOPE

Interface for the Apple II Computer



The APPLESCOPE system combines two high speed analog to digital converters and a digital control board with the high resolution graphics capabilities of the Apple II computer to create a digital storage oscilloscope. Signal trace parameters are entered through the keyboard to operational software provided in PROM on the DI control board.

- DC to 3.5 Mhz sample rate with 1024 byte buffer memory
- Pretrigger Viewing
- Programmable Scale Select
- Continuous and Single Sweep Modes
- Single or Dual Channel Trace
- Greater than or less than trigger threshold detection

Price for the two board Applescope system is **$595**
EXTERNAL TRIGGER ADDAPTER **$29**

APPLESCOPE EXPANSION CAPABILITY

APPLESCOPE - HR12 High resolution 12 bit analog to digital converter with sample rates to 100 Khz. Software included on disk. Price per channel **$695**

APPLESCOPE - HRHS High Resolution AND High Speed. Combines two 6 bit flash Analog to Digital converters to give 10 bit converter accuracy at a maximum 7 Mhz. sampling rate. Software included on disk. Price per channel **$695**

SCOPE DRIVER Advanced software for the Applescope system provided on 5¼" floppy disk. Available options include:

- Signal Averaging - Acquires 1 to 255 signal sweeps and displays the averaged result.
- Digital Volt Meter - Allows use as a real time DVM or use to measure points on an acquired sweep.
- Hard Copy - Uses graphics printer to produce hardcopy output of displayed traces.
- Disk Storage - Allows automatic storage and recover of acquired data on floppy disks.
- Spectrum Analyzer - Calculates and displays frequency spectrum of acquired data.

The basic SCOPE DRIVER package cost is **$49** plus **$10** for each selected option.

BUS RIDER
LOGIC ANALYZER for the APPLE II

The BUS RIDER circuit card silently rides the Apple II peripherial bus and allows real time tracking of program flow. Software provided on EPROM allows set up of trace parameters from the keyboard and read back of disassembled code after a program has been tracked.

- 32 bit by 1024 sample memory buffer
- Monitors Data and Address bus plus 8 external inputs
- Trigger on any 32 bit word or external trigger
- Pretrigger viewing

The BUS RIDER is an invaluable development tool for anyone working with Apple II or Apple II+ computers. Price **$295**

Apple II BUS EXTENDERS **$19.95**
Allow easy access to Apple II peripherial circuit cards.

SCOPE PROBES 100 Mhz. Bandwidth X1 & X10 switch selectable oscilloscope probes. Price each **$49.95**

For further information contact:

VISA Master Charge

Dealer Inquiries Invited

RC ELECTRONICS INC.
7265 Tuolumne Street
Goleta, CA 93117
(805) 968-6614

Figure 8. A Graph of R = SQR(3600/THETA).

R = SQR(3600/THETA)

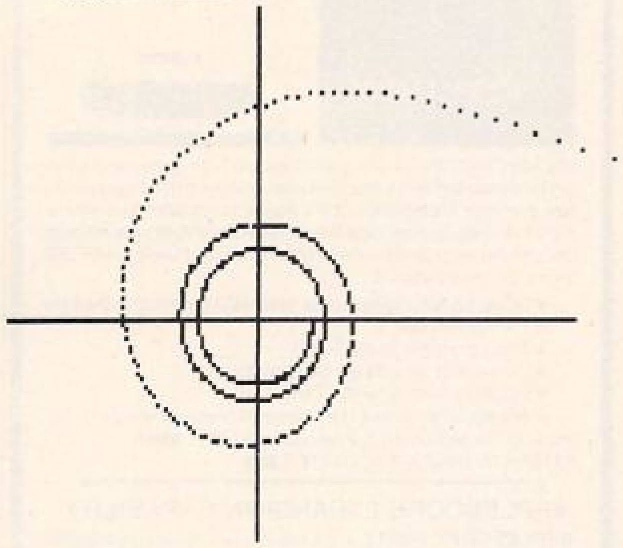
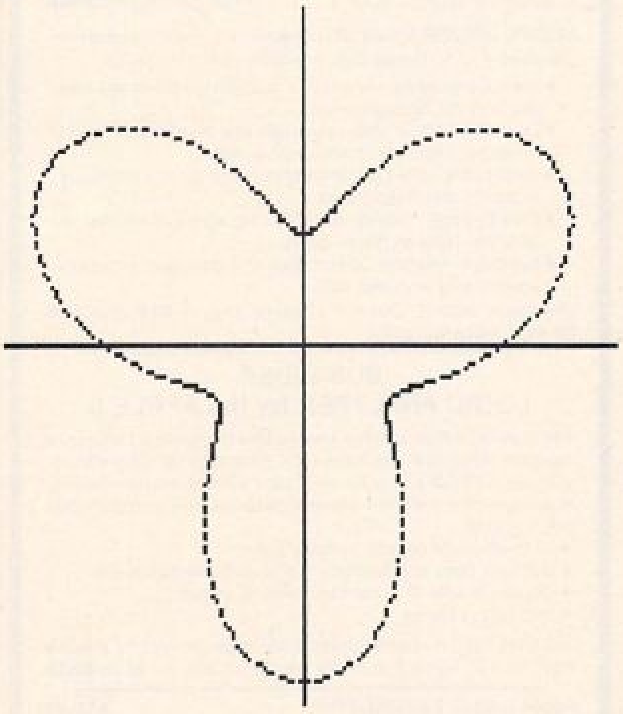


Figure 9.
Untitled Graph with R = 25*(2 + SIN(3*THETA)).

R = 25*(2 + SIN(3*THETA))



Program 1. A Simple Program to Graph Polar Functions

```
←LIST

10  INPUT AA,AB
20 ANG = AA
30  HGR
60 THETA = 3.1415926 * ANG / 180
70 R = 85 *  SIN (2 * THETA)
80  PRINT R,ANG
90 X = 85 + R *  COS (THETA)
100 Y = 85 - R *  SIN (THETA)
120  HPLOT X,Y
130 ANG = ANG + 1
140  IF ANG <  = AB THEN 60
150  END
```

Program 2. An Elaboration of Program 1.

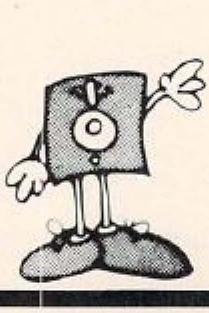
```
←LIST

10  INPUT AA,AB
20 ANG = AA
30  HGR2
40  HPLOT 1,80 TO 160,80
50  HPLOT 80,1 TO 80,160
60 THETA = 3.1415926 * ANG / 180
70 R = 80 *  SIN (3 * THETA)
80 X = 80 + R *  COS (THETA)
90 X =  INT (X + .5)
100 Y = 80 - R *  SIN (THETA)
110 Y =  INT (Y + .5)
120  HPLOT X,Y
130 ANG = ANG + 1
140  IF ANG <  = AB THEN 60
150  END
```

©

MEMOREX
FLEXIBLE DISCS

WE WILL NOT BE UNDERSOLD!! Call Free **(800)235-4137** for prices and information. Dealer inquiries invited and C.O.D.'s accepted.

PACIFIC EXCHANGES
100 Foothill Blvd.
San Luis Obispo, CA
93401. In Cal. call
(800)592-5935 or
(805)543-1037



VISA

ECHO][™ SPEECH SYNTHESIZER
UNLIMITED VOCABULARY AT YOUR FINGERTIPS

Give your Apple* something to talk about with an ECHO][Speech Synthesizer. The ECHO][offers intelligible voice-output while using a minimum of RAM. The ECHO][uses LPC technology pioneered by Texas Instruments, coupled with a phoneme-based operating system allowing you to create any vocabulary desired. Variable stress, pitch and volume let your Apple ask questions or make exclamations while also allowing for optimal quality. Speech can easily be added to BASIC programs with PRINT statements.

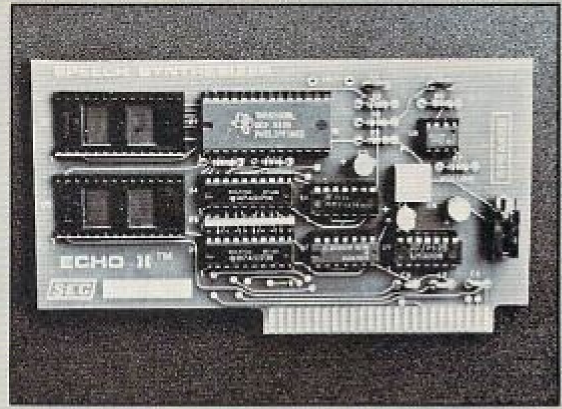
The applications of synthesized speech are limited only by your imagination. From educational programs to games to business applications, speech-output adds a new dimension to your Apple's capabilities.

The ECHO][Speech Synthesizer comes complete with speaker and cable, instruction manual, speech editing features and a sample vocabulary. The ECHO][requires 48K, Applesoft and at least one disk drive. Suggested list price is $225.

For further information, contact your dealer or Street Electronics Corporation.

Dealer inquiries welcome.

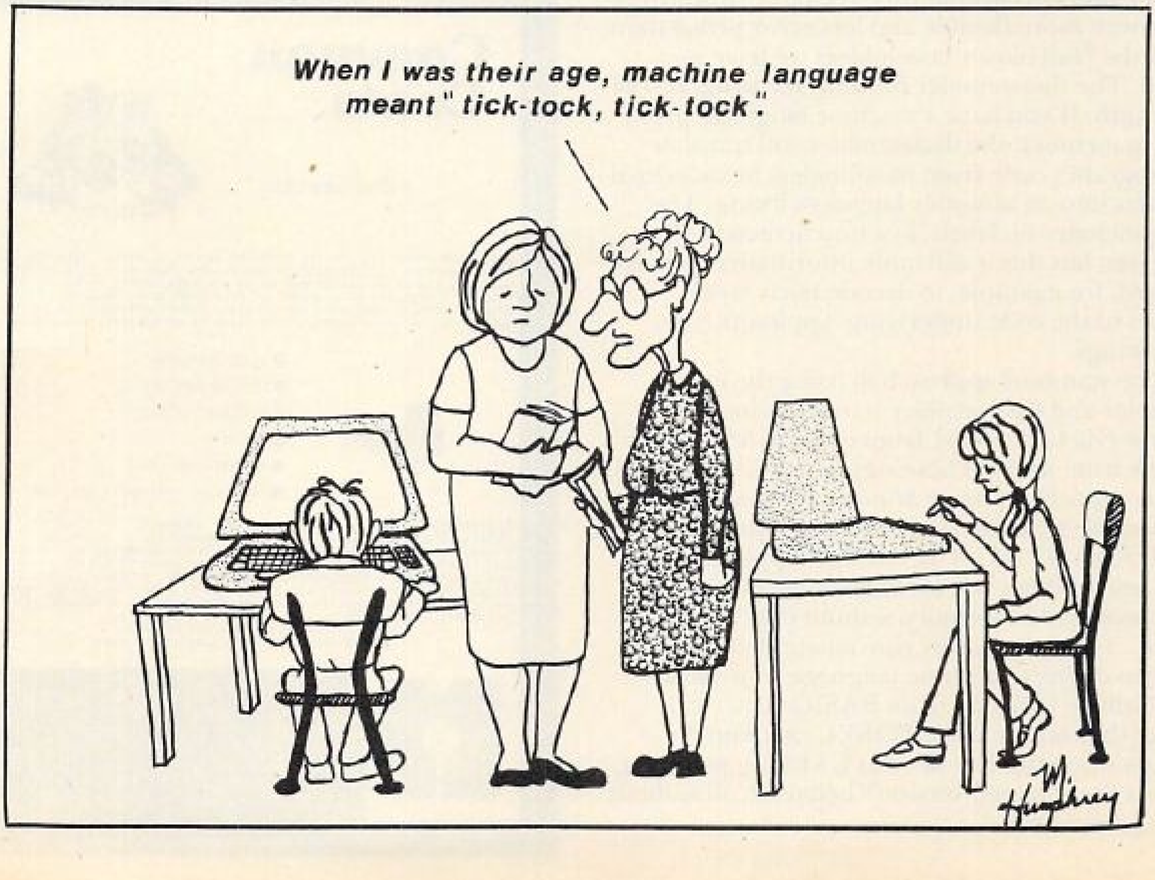
* Trademark of Apple Computer Company



SEC STREET ELECTRONICS CORPORATION

3152 E. La Palma Ave., Suite D
Anaheim, CA 92806 (714) 632-9950

COMPUTER CLASS M. HUMPHREY


When I was their age, machine language meant " tick-tock, tick-tock ".

Disassembling Machine Language Programs Without Leaving BASIC

John R. Vokey and H. Cem Kaner
McMaster University
Hamilton, Canada

One of the nice features of the Apple computer is that it has a built in mini-assembler and disassembler. The mini-assembler is all you need for entry of short machine language programs. In fact, for short programs, this free piece of software has proven more flexible and less error prone than two of the "full blown" assemblers we have purchased. The disassembler is useful for programs of any length. If you have a machine language program in memory, the disassembler will translate the program's code from meaningless hexadecimal numbers into an assembly language listing. The listing includes no labels, just instructions and addresses, but this is still quite informative. It is not too hard, for example, to decode fairly large sections of the code underlying Applesoft from such listings.

The standard approach to using the mini-assembler and disassembler is to jump into the monitor (via CALL -151 from either BASIC) and to work from there. These steps are well described in your *Apple II Reference Manual.* However, it is also possible to access some of these monitor commands from BASIC. The one line Applesoft program below allows you to disassemble machine code anywhere in memory without ever leaving BASIC. This is especially convenient if you are trying to debug a machine language subroutine which will be CALLed from BASIC. You can change the routine using POKEs, examine the changes using this line in your CALLing program, and test the changed version's behavior, all without leaving Applesoft.

The program works by passing the user-specified START location of the code to be disassembled to the monitor program counter (labelled PC in the program). It then calls the monitor LIST subroutine which we label disassemble in the program. This routine disassembles the next 20 lines of machine code, incrementing the monitor program counter locations appropriately, and returns control to BASIC. The BASIC program then compares the value of the monitor program counter to the user specified value FINISH. If there is more to be done before location FINISH is reached, the program waits until you press any key, then continues the listing. Once FINISH is reached, the program ends.

As an example of the use of the program, if you set START to 65118 and set FINISH to 65140, you will disassemble the disassembler.

```
1000 DISASSEMBLE = 65121: PC = 58:
POKE PC, START - INT (START / 256)
* 256: POKE PC + 1,START / 256: FOR
I = 0 TO 1: HOME: CALL DISASSEMBLE:
PRINT: PRINT TAB (13);"<PRESS ANY
KEY>": GET Z$: I = ( PEEK (PC + 1)
* 256 + PEEK (PC)) > FINISH: NEXT I
```

©

Common Cents

FORECASTER

With your computer, you can forecast the stock market like the Wall Street experts. FORECASTER, a mutual and money market fund tracker, monitors the stock market and helps you make timely decisions to achieve a high and safe growth of your investment dollars. Our program tracks, estimates and predicts future movements in the stock market and selected mutual funds based on an extensive data base.

- ● AVOID CONFUSION
- ● ACHIEVE HIGH YIELDS
- ● PRESERVE CAPITAL
- ● LOW RISK
- ● TOP INTEREST RATES
- ● TAKES LESS THAN ½HOUR PER WEEK

FORECASTER is written in simple, clear language. No advanced knowledge of mathematics, computer programming, or investing experience is required. Send $10.00* for programming report and user's manual, or $75.00* for complete software package. Please specify Apple II+ (Applesoft), CP/M Basic 80, or TRS 80 Level II. (Apple II+ and Applesoft are trademarks of Apple Computers, CP/M is a trademark of Digital Research, Basic 80 is a trademark of MicroSoft, and TRS 80 is a trademark of Tandy Corp.
* Add $2.00 for postage and handling
New Mexico residents also add 4% tax

META SOFTWARE ENGINEERING
P. O. Box 18056
Albuquerque, NM 87115
D 1

Named GOSUB With Variable Passing

Mike Smith
Calgary, Canada

In **COMPUTE!** # 12, I described a machine language program which would allow subroutines to be called by name rather than by number. This article is an extension of that idea. It describes a machine language program which allows parameters to be passed in and out of subroutines.

One of the nicer features of FORTRAN and PASCAL is their ability to pass variables into a subroutine. This feature is very useful when you wish to do the same operation on a large number of variables. Passing parameters into subroutines is convenient since the variable names used outside the subroutine don't have to be the same as used for the calculation within the subroutine. This makes programming and documentation easier. In addition, subroutines of this type can be used as a sort of multi-line function.

A Brief Example Of Parameter Passing

Suppose that you wish to perform a complicated operation upon variables A, B and C and have the answer returned in D. Then you wish to have the same operation performed upon the variables A1, B1 and C1 and have that answer returned in D1.

In FORTRAN that program would look like this:

```
CALL COMPL(A,B,C,D)
     (call subroutine with first variable set)
CALL COMPL(A1,B1,C1,D1)
     (then with the second set)
.........
(Use D and D1 in calculations)
.........
SUBROUTINE COMPL(W,X,Y,Z)
     (use dummy variables with subroutine)
(Complicated calculation using W, X and Y)
.........
Z = .....
RETURN
```

In Applesoft BASIC things are a little more difficult. First, you must call the subroutine by a *number* rather than by a *name*. A second problem is that you can't pass the names of variables into the subroutine. Instead, you must move (reassign) the values into the variable names used in the subroutine. An equivalent Applesoft BASIC program would look something like:

```
10  W=A : X=B : Y=C
     (reassign first set of variables)
20 GOSUB 1000
30 D=Z
40 W=A1 : X=B1 : Y=C1
     (reassign second set)
50 GOSUB 1000
60 D1=Z
70 .........
    (Use D and D1 in calculations)
    .........

1000 (Complicated calculation using W, X and Y)
     .........
1100 Z= .....
1110 RETURN
```

Having to remember the subroutine number is no great problem if you are the person who did the programming, provided you only did the programming a week or so ago, and have not yet forgotten what subroutine number was needed for what. Having to reassign variables, as in statement 40, is no great problem either, provided you don't have a large number of different variables that need to be worked on. But why do something that the computer can make easier to understand and do?

The program described in this article uses the Applesoft BASIC *ampersand* command (&) to allow the naming of subroutines and the easy passing of numerical data. With the machine code routine installed in memory, the Applesoft program above becomes:

```
10  COMPL = 1000
     (establish the subroutines name)
20  & GOSUB COMPL !COMPL(0),A,B,C,D!
     (pass the parameters)
30  & GOSUB COMPL !COMPL(0),A1,B1,C1,D1!
40  .........
    (Use D and D1 in calculations)
    .........

1000  & GET !COMPL(0),W,X,Y,Z!
      (identify the dummy variables)
1010  (Complicated calculation using W, X and Y)
      .........
1100  Z = ....
1110  & RETURN !COMPL(0)!
```

In addition to passing parameters, Applesoft will now support GOTO and GOSUB statements that have names instead of numbers. For example

```
   JUMP = 1000 : & GOTO JUMP or COMPL = 1000 :
     & GOSUB COMPL
FIRST = 1000 : DEUX = 2000 : ON X GOSUB FIRST,
     DEUX
```

I decided to develop this parameter passing routine because I am repeatedly asked to translate FORTRAN program with subroutines into Applesoft. Most of those subroutines pass variables. Making sure that I didn't duplicate names and that

I reassigned the right variable, was too much of a hassle. Hence this routine.

Loading The Program

The machine language program as described in this article is too long to put in a normally unused area of memory. The cassette buffer (at $300) will only accept around $CF locations before running into the DOS pointers at $3D0.

The program could be placed high in memory, just below the normal HIMEM. The HIMEM pointers must then be adjusted so that the program is not touched by Applesoft when strings are used. However, this means that people using 48K and 32K Apples, with or without the Program Line Editor at the top of memory, will all need different programs. The modifications are simple, if you know how. Therefore, I have adopted the technique of moving LOMEM up $200 bytes and storing the machine language code in the space created. Then everybody gets the same code.

Before entering the demonstration BASIC program, type:

POKE 104,10 : POKE 2560,0 : NEW

These three instructions adjust LOMEM and the various Applesoft RUN, LOAD or SAVE programs. The pointers can be shifted down to their normal place by typing FP.

After the BASIC program has been run, the machine code can be saved by the command BSAVE VARIABLE.PASS, A$803,L$181. The program will stay active, below your BASIC program, until you power down or do an FP.

To reload the ML program the next time you power up, type BRUN VARIABLE.PASS either from the keyboard or as part of your HELLO program. The LAST line of the HELLO program should be PRINT CHR$(4);"BRUN VARIABLE.PASS".

The first couple of statements of the hex code are the machine language equivalent of POKE 104,10 : POKE 2560,0 : NEW. That means that you only have to adjust the memory the first time you enter in the code. If you forget to adjust the memory before running the demonstration BASIC program, you will receive the message SYNTAX ERROR in 34057, a non-existent line. Simply type NEW : POKE 104,10 : POKE 2560,0 : NEW, reload the program from disk and RUN again. If you didn't adjust LOMEM, then, when the BASIC program stored the machine language program, it did so all over itself, causing a gigantic mess.

There is a sneaky reason for starting the machine language program at $803 (2051) rather than at $800, the start of the empty memory area. Suppose that, for some reason or another, you need to enter FP to recover from your program doing something strange. Typing FP causes 0's to be written at locations $800-$802 to indicate that there is no longer a program in the memory. This misses the ML program since it starts at $803. Thus, a quick CALL 2051 and ABRACADABRA, the pointers shift and the program is back in business.

The details of the demonstration and machine language programs are given after the description of the new SYNTAX of the instructions and limitations of the new commands.

Syntax For The New Commands

& GOSUB NAME !NAME(0),A,B,.....!

The name of the subroutine must be predefined before the subroutine is called (e.g. NAME = 1000).

The first parameter after the exclamation mark *must* be an array; otherwise, a BAD SUBSCRIPT ERROR occurs. It is suggested that the name of this array be the same as the name of the subroutine; for ease of remembering rather than necessity. If more than ten parameters are to be passed by the routine, the array must be DIMensioned to the number of parameters. No check is performed to see if the array is large enough for all the parameters used.

The other parameters must be numerical, either real variables (A, B etc.) or elements of a real array (A(1), B(1) etc.). The arrays don't have to be predimensioned unless their length is greater than ten. Errors will occur on attempting to pass a string (TYPE MISMATCH) or an integer (SYNTAX). It should be noted that it is the *value* of the array element that is moved and *not* the array itself. This means that you can't pass over the whole array by passing over the first element of an array. (c.f. In FORTRAN, it is the address which is passed and not the value of the array element. So, the whole array can be accessed from FORTRAN subroutine if you know the first address. In Applesoft, memory is continually being repositioned. The address of any variable is therefore continually changing, making any address stored very quickly invalid.)

The parameters do not need to have been defined before calling the subroutine. The machine language program makes use of Applesoft routines which automatically allocate space in the memory for new arrays and variables.

& GET !NAME(0),P,Q,.....!

This should be the first statement of the subroutine. The subroutine can't be recursive (it can't call itself).

This command does not extend an existing Applesoft command as did the & GOSUB, & RETURN and & GOTO commands. Therefore I had to use a different command. I decided to use GET, Since to me, this new command goes and *gets* the parameter values. If you would prefer a different command, such as LOAD, then the modification to allow this is simple. To have a different command, POKE its token into location 2600 ($828) before BSAVEing the program. For example, POKE 2600 ,167 will change this command to be & RECALL !......! rather than & GET !......!. (See page 121 of the *Applesoft Manual* for a list of the tokens).

The first parameter after the exclamation mark must be the same array used in the & GOSUB statement, otherwise unexpected values will be put into the parameters (P etc).

The other parameters must be real, otherwise a TYPE MISMATCH or SYNTAX ERROR will result. Either real variables (P) or elements of real arrays (P(1)) may be used. Again, the parameters don't have to be predefined before the subroutine call, unless they are arrays of length greater than ten. If the arrays need to be DIMensioned remember to do it *outside* the subroutine. Otherwise a REDIMENSIONED ARRAY ERROR will result on the second subroutine call.

The number of parameters in the & GET statement should be the same as the number of parameters in the & GOSUB statement. If this condition is not met, strange values could arrive in the parameters of the & GET statement.

& RETURN !NAME(0)!

The array used in the & RETURN statement should be the same array as used in the & GOSUB and & GET statements. As this array is used to temporarily store text pointers to the & GOSUB and & GET statements, strange results could result if the wrong array is used. However, it is probable that, instead of funny results, a SYNTAX ERROR will occur. The likelihood of the wrong array pointing to valid names in separate locations in memory is very small.

If the number of parameters in the & GET statement is not the same as the number of parameters in the & GOSUB statement, unpredictable values will be put into the parameters.

& GOTO NAME and & GOSUB NAME

The name of the subroutine must be established before it is called. If these commands are used, a normal RETURN is all that is needed. If & GET and & RETURN are used, a SYNTAX ERROR will occur.

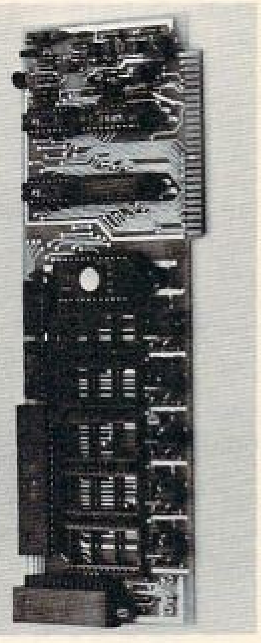
& ON X GOSUB FNAME, SNAME and & ON X GOTO FNAME, SNAME

These ON X.... commands are supported, provided that no parameters are passed. That means that & ON X GOSUB FNAME, SNAME is permitted but & ON X GOSUB FNAME !FNAME(0),A,B,C,D!, SNAME !SNAME(0),A,B,C,D! is not. I felt that passing parameters in ON X... statements made

OMNI

The "Everything" I/O Board for the Apple II & II+

OMNI is a multi-function input/output board for the Apple II or II+ computer. It provides, on a single board, most of the "missing" features needed to make the Apple a complete computer. With OMNI your Apple can have:



- Parallel I/O with handshaking
- RS232 Level serial I/O (software driven)
- 24 Hr. Real Time Clock with Alarm
- 2K EPROM with graphics, I/O driver, and screen editor firmware
- 256 Byte PROM supervisory firmware
- Six 2K PROM/EPROM expansion sockets software selectable (one socket used for Eclectic firmware, 5 available for user)

INPUT:
- Generate full ASCII character set from keyboard
- Optional shift key detection
- User-definable "soft" keys with screen legends
- Integrated text line editor full cursor movement, insertion/deletion modes
- Key legend stickers included
- Demonstration Diskette with programming examples and a Soft Character Editor

OUTPUT:
- Full 96 character ASCII display
- Concurrent 64 user-definable "soft" characters
- Optional character overstrike and EOR on background
- Optional double-width color characters
- Character rotation in 90° steps

Never before have so many functions been available on a single board.

OMNI was designed with one major goal in mind, flexibility. The OMNI system consists of some extremely simple but very sophisticated hardware, a large amount of powerful firmware (programs permanently residing in Read Only Memory chips), and an equally extensive amount of software (programs residing on diskette that are loaded into RAM as needed). In addition, OMNI comes with extensive documentation.

ECLECTIC SYSTEMS CORPORATION

To order TOLL FREE: 1-800- 527-3135
or Order by Mail Below

Eclectic Systems Corporation, P.O. Box 1166, Addison, TX 75001

Here's my order for OMNI at $268 plus $3 for shipping and handling (UPS surface, unless specified), $5 overseas.

☐ My check is enclosed.

☐ Please charge my VISA ____ Mastercard ____

Account # __________ Expires __________

Name ______________________

Address ______________________

City/State __________ Zip __________

Signature ______________________

the statements very unwieldy. The original idea behind introducing these new commands was to make the programs more readable rather than less. Multiline IF...THEN commands would do the same job, in a more readable fashion. For those people interested in implementing the unweildy ON... version, I have included the additional code needed (lines 176-189).

Warning

Warning on renumbering and crunching programs: Renumbering programs will not change the values of variables. Therefore, they will not change the pointers to the subroutines called by these new commands. This must be done by hand after the renumbering is complete. Utilities that crunch programs will not recognize the fact that the subroutines are being called and therefore will remove them as dead code. To overcome this removal problem, a dummy line that calls all subroutines, must be added to the program. After crunching, delete the dummy line. For example:

```
10 NAME=1000 : FIRST=2000
   (define the subroutines)
20 IF X=0 THEN GOSUB 1000 : GOSUB 2000 :
   GOTO 20 (dummy line to be removed after
   crunching)
```

Note that the dummy line is an IF..THEN statement that loops to itself. This means that a CRUNCHER, such as the one in DAKIN 5 PROGRAMMING AIDS 3.3, will leave that line alone, making it easy to remove.

BASIC Program Description

Line 180 – Establishes the machine language program.

Line 200 – Establishes the name of the subroutines to be called.

Line 220 – Demonstrates the command & GOSUB without passing any variables.

Line 250 – A loop is used to show that the stack is not corrupted by using these new commands. An OUT OF MEMORY ERROR will occur for 25 GOSUB calls without a proper return.

Line 260-280 – Establish random numbers for use in the variables.

Lines 290-320 – Demonstrates the & GOSUB command using both simple variables and arrays elements. The example subroutine adds together the first two numbers passed to it. The result is passed back in the third parameter.

Line 360 – Demonstrates that the subroutine call operated and that parameters were passed both ways.

Line 370 – Delay loop.

Line 1000 – Subroutine called without passing variables.

Line 2000 – New subroutine showing that variables were passed and used within the subroutine.

Line 5000-5070 – Machine language loading subroutine. It first checks that the DATA statements have been typed in correctly. Each DATA statement is the value of 16 locations plus the sum of the previous 16 locations used as a simple checksum. A typo error is indicated if the checksum is not the sum of the previous 16 locations.

Line 5080-5120 – Checks that POKEs have been performed.

Line 5130-5140 – POKEs the routine into memory.

Line 5150 – This establishes the AMPERSAND vector (&) pointers. This call is not necessary if the machine code is BRUN, but is necessary if the subroutine is BLOADed. Note that the CALL from BASIC is not the start of the ML program. If we did CALL the start of the program, an automatic NEW would occur, wiping out the demonstration program.

Machine Code Description

Briefly, the machine language program works as follows:

& GOSUB NAME!NAME(0),A,B...! The text pointers to the variable A are stored in the first two bytes of NAME(0). Then the value of A is moved into NAME(1), B into NAME(2) and so on.

& GET !NAME(0),W,X,...! The text pointers to the variable W are stored in the second two bytes of NAME(0). The value of NAME(1) is moved into W, NAME(2) into X and so on.

& RETURN !NAME(0)! The text pointer to W are recovered. The current values of W, X .. are moved into NAME(1), NAME(2) etc. Then the text pointer to A is recovered. The values in NAME(1), NAME(2) ... are moved into A, B....

The method of implementing the other commands is described in **COMPUTE!** #12.

Lines 15-31 – Zero page usage.

Lines 33-43 – Definition of tokens.

Lines 45-61 – Pointers to Applesoft routines. Internal Applesoft routines are used to cut down the amount of code required.

ADJMEM and AMPER. **Lines 65-77** – Do the machine language equivalent of POKE 104,0 : POKE 2560,0 : NEW. Then set the AMPERSAND vector.

ENTRY. **Lines 80-92** – Check on which of the new commands is required.

GOTO. **Lines 94-99** – Front end of the normal Applesoft GOTO routine moved and modified to allow variables and numbers to be used in the GOTO statement.

GOSUB. **Lines 101-134** – Handling of the & GOSUB command.

Line 101 – Front end of the normal Applesoft GOSUB routine moved and modified to allow

utility city

21 of our most-asked-for Apple Utilities

by Bert Kersey 48K Applesoft ROM

For YOUR Big Apple—Our 21 most-asked-for Apple Utilities on one big disk—**List Formatter** makes properly spaced & indented listings with printer page breaks. Each program statement is on a new line with If-Then's & Loops called out; a great de-bugger! **Catalog in any number of columns** & any page-width to CRT or printer. Automatically post the Run-Number & last-used Date in your programs. **Make any command invisible** in your listings; Access program lines in memory for garbage repair & "illegal" alteration; Quickly sort & store info on disk; Run any Applesoft file while another stays intact; Move lines within your programs; Renumber to 65535; Save inverse, INVISIBLE & flick file names; Convert dec to hex & binary, or INT to FP; Append programs; dump the text screen to ANY printer . . . More too: **21 Programs Total!**

PLUS . . . APPLE TIP BOOK NUMBER THREE!
40 PAGES of new tips, tricks & articles—"DOS Trickery," "Copy Stoppers," "Programming the Reset Key" & much more. Plus understandable explanations of how each U-City Program works—One of the best Apple learning tools on the market today!

ALL 21 PROGRAMS **$29.50** +$1.50 shipping
- ✓ **Utility City** on Applesoft Disk
- ✓ 40-page Beagle Bros Apple Tip Book # 3
- ✓ Apple PEEKS, POKES & POINTERS Chart

alpha plot

Hi-Res Apple Graphics/Text Utility
by Bert Kersey & Jack Cassidy 48K REQUIRED

HI-RES DRAWING: Create hi-res pictures & charts, **appendable to your programs.** Keyboard or Paddle control; Optional Xdraw Cursor (see lines before you draw!); Any color mix or REVERSE (opposite of background); Circles, Boxes & Ellipses, filled or not. Bonus Programs too—**SCRUNCHER** stores hi-res in as little as 1/3 normal disk space. **SHIFTER** transfers any portion of the hi-res screen. Also superimpose hi-res images and convert Hi-Res to Lo-Res & back for fascinating abstracts!

HI-RES TEXT: Beautiful upper & lower case with Descenders; color or reverse; Positionable anywhere (NOT restricted by Htabs & Vtabs). Professional looking **PROPORTIONAL SPACING!** Adjustable Type Size, Leading (line spacing) & Kerning (letter spacing). Multi-directional typing: up, down, even backwards!

PLUS . . . APPLE TIP BOOK NUMBER FOUR!
40 PAGES of tips for making the most of your Apple's advanced graphics capabilities. Alpha instructions plus shape table tips, new titling & color tricks and fascinating animation experiments!

$39.50 +$1.50 shipping
- ✓ **Alpha Plot** on Applesoft Disk.
- ✓ 40-page Beagle Bros Apple Tip Book # 4
- ✓ Apple PEEKS, POKES & POINTERS Chart.

NEW Apple® Utility Disks!
With Free PEEKS & POKES Wall Chart!

Beagle Bros MICRO SOFTWARE
4315 Sierra Vista
San Diego,
CA 92103
714-296-6400

BONUS WITH EACH DISK!
POKE your Apple all night long with this handy 11 × 17 reference poster! The most useable PEEKS, POKES, POINTERS & CALLS scrounged up from every source imaginable! A great companion to our original Apple Command Chart.

WITH EACH ORDER, YOU WILL RECEIVE . . .
- ☐ **Dos Boss, Utility City** or **Alpha Plot** on Applesoft Disk
- ☐ A 40-page **Apple Tip Book** (each entirely different)
- ☐ An 11x17 Apple **PEEKS, POKES & POINTERS** Chart

dos boss
DISK COMMAND EDITOR
by Bert Kersey & Jack Cassidy

A classic utility you will ENJOY! **Rename Commands:** "Catalog" can be "Cat," etc. **Save-Protect your programs;** Unauthorized copy attempts produce "Not Copyable" message. 1-Key Program Selection from catalog (extremely handy!). Catalog Customizer—Change Disk Volume message to your title; Omit or alter file codes; Catalog by file type, etc. Rewrite Error Messages—"Syntax Error" can be "Oops!" or anything you want!

All of Dos Boss's change features may be appended to your programs, so that anyone using your disks on any Apple (booted or not) will be formatting DOS the way you designed it!

PLUS . . . APPLE TIP BOOK NUMBER TWO!
With valuable DOS information: "Poking Around in DOS," "Making Programs Unlistable," "Two-Sided Disk Tips," "Care of Your Drives," "Creative Peeking & Poking," "3.2 vs 3.3," etc.

ONLY $24. +$1.50 shipping
- ✓ **Dos Boss** on Applesoft Disk
- ✓ 40-page Beagle Bros Apple Tip Book #2
- ✓ Apple PEEKS, POKES & POINTERS Chart

MINIMUM 32K REQUIRED

AT YOUR APPLE DEALER NOW, OR ORDER BY PHONE AT ANY HOUR... TOLL FREE!

Ask around about Beagle Bros Software.
Our programs are practical, friendly & user-oriented, and always come with an amazing amount of documentation & juicy Apple information. If your dealer doesn't have our stuff, get on his case about it, or order directly from us. We ship all disks & books WITHIN 48 HOURS.

24 hour TOLL FREE Order Desk:
MasterCard, VISA or COD orders, call:

Nationwide:	**800-854-2003**	ext. 827
California:	**800-522-1500**	ext. 827
Alaska/Hawaii:	**800-854-2622**	ext. 827

(Orders only please. Questions, phone 714-296-6400.) COD orders, add $3.)

Or order by mail (use this coupon or separate sheet)

(use this coupon or separate sheet)

RUSH! The disks checked below
Plus The Tip Book and PEEKS & POKES Chart!

☐ **UTILITY CITY** ($31) ☐ **ALPHA PLOT** ($41) ☐ **DOS BOSS** ($25.50)

(Prices include First Class Shipping; Calif. residents add 6%.)

Specify

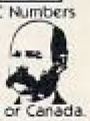
NAME$= ____________ ☐ **3.2** ☐ **3.3**

ADDR$= ____________

CITY$= ____________ ZIP= ____________

VISA **VISA/MASTERCARD: Include Account No. & Expiration Date.** master charge

Mail U.S. check, money order or Visa/MC Numbers to: **BEAGLE BROS, Dept. 5**
4315 Sierra Vista,
San Diego, CA 92103
Telephone (714) 296-6400



Please add $4.50 for shipping outside US or Canada.
COD orders, add $3 California residents, add 6%.

---

variables and numbers in subroutine calls.

Line 121 – Is the first parameter an array? This array is used for the storage of the text pointers and the parameters. The stack would get too full if it were used.

Line 130: – Store text pointers.

Line 132 – Move the other parameters into the array for storage.

GET. **Lines 136-140** – Handling of the & GET command.

Line 136 – Locate the storage array.

Line 137 – Store the text pointers.

Line 139 – Move values stored in the array into the new parameters.

ARRay-GET. **Lines 142-149** – Gets and stores the location of the storage array after checking the leading exclamation mark.

ON. **Lines 151-189** – Handling of the ON X... command.

Line 152 – Get the value of X.

Line 154 – Determine if ON..GOTO or ON.. GOSUB.

Line 163 – Decrement X until find the subroutine requested.

Line 167 – Step over the values not being used.

Line 172 – Return to BASIC if subroutine not found.

Line 176-189 – Adding these instructions will allow the passing of parameters in ON X... commands.

RETURN. **Lines 191-213** – Handling of the & RETURN command.

Line 191 – Locate the storage array.

Line 198 – Store the current text pointers.

Line 199 – Recover the text pointers from the & GET statement.

Line 200 – Move the current values of the parameters in the & GET statement into storage.

Line 201 – Reset the storage array pointers.

Line 205 – Recover the text pointers from the & GOSUB statement.

Line 206 – Move the values in the storage array into the parameters used in the & GOSUB statement.

Line 207 – Recover the current text pointers and perform a normal RETURN.

CHecK-ARRay. **Lines 215-228** – Checks and adjusts the pointers to the storage array if new variables have been introduced during the commands & GOSUB and & GET.

Modifications to the next two subroutines, PARSTO and STOPAR will allow the passing of INTEGER parameters.

PARameters-to-STOrage. **Lines 230-243** – Moves the current values of the parameters in the & GOSUB and & GET commands into the storage array. Checks for integers and strings.

STOrage-to-PARameters. **Lines 244-257** – Moves the values in the storage array into the parameters in the & GOSUB and & GET commands.

STOre-TeXT-pointers. **Lines 259-264** – Stores the current text pointers into the zeroth element (NAME(0)) of the storage array. The Y register is preset.

GET-TeXT-pointers. **Lines 266-271** – Recovers the text pointers stored in the zeroth element of the storage array according to the value set in the Y register.

ADJust-PoinTers. **Lines 273-276** – Adjust the pointers to the storage array if they have shifted because a new variable has been made. Note that the pointers don't have to be adjusted if a new array has been made. All new arrays will be placed above the storage array in memory as the storage array is defined first.

COMmand-END. **Lines 278-283** – Looks for the final exclamation mark (!) of the command or other parameter. Pops the last subroutine address off of the stack allowing a quick return to BASIC if at the command's end.

References

"Applesoft Internal Entry Points" by Applesoft Computer Inc. in Apple Orchard March/April 1980, p. 12.

"Some Routines in Applesoft Basic" by J. Butterfield in COMPUTE!, September/October 1980, p. 68.

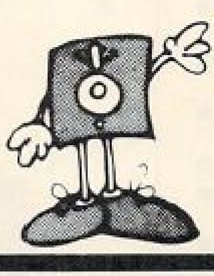
"Resolving Applesoft and Hires Graphics Memory Conflicts" by J. Schmoyer in COMPUTE!, April 1981, p. 76.

"Using Named GOSUB and GOTO Statements in Applesoft BASIC" by M. Smith in COMPUTE!, May 1981, p. 64.

```
100 ********************************
110 REM * MIKE SMITH              *
120 REM * 304, 86TH AVENUE SE     *
130 REM * CALGARY, ALBERTA        *
140 REM * CANADA  T2H 1N7         *
150 ********************************
160 REM
170 REM SET UP THE MACHINE CODE
180 GOSUB 5000
190 REM   SET UP THE SUBROUTINE NAM
    ES
200 DEMO =1000:ADDIT = 2000
210 REM  DEMONSTRATE NAMED GOSUB AN
    D GOTO
220 &  GOSUB DEMO:JUMP = 240: &  GO
    TO JUMP
230 REM DEMONSTRATE STACK OKAY
240 PRINT "HERE BY NAMED GOTO": PRI
    NT
250 FOR J = 1 TO 25
260 REM MAKE UP NUMBERS
270 K = INT (10 * RND (1)): = INT (
    10 *  RND (1))
280 P = INT ( 10 * RND (1)):Q(1) = 
    INT (10 * RND (1))
290 REM
300 &  GOSUB ADDIT!ADDIT(0),K,L,M!
310 REM   DEMONSTRATE PASSING OF AR
    RAY ELEMENT
320 &  GOSUB ADDIT!ADDIT(0),P,Q(1),
    R!
330 REM
340 REM PRINT AND SHOW THAT HAVE US
    ED SUBROUTINE
350 REM
360 PRINT K;" + ";L;" = ";M: PRINT 
    P;" + ";Q(1);" = ";R: PRIN
    T
370 FOR Z = 1 TO 500: NEXT Z
380 NEXT J: STOP
970 REM
980 REM  DEMONSTRATION SUBROUTINE
990 REM
1000 PRINT : PRINT "HERE BY THE GOSU
    B CALLED DEMO"
1010 PRINT : RETURN
1960 REM
1970 REM  SUBROUTINE ADDIT
1980 REM
1990 REM  DEMONSTRATE PASSING BACK O
    F ARRAY ELEMENT
2000 &  GET !ADDIT(0),T,U,V(4)!
2010 V(4) = T + U
2020 &  RETURN !ADDIT(0)!
4970 REM
4980 REM  MACHINE CODE ESTABLISHED
4990 REM
5000 BOT = 8 * 256 + 3:HIGH = 9 * 25
    6 + 10 * 16 + 2
5010 REM FLAG FOR CHECKSUM
5020 OK = 1:LINE = 6000
5030 FOR J = BOT TO HIGH STEP 16
5040 CHECK = 0: FOR K = J TO J + 15:
     READ IT:CHECK = CHECK + I
    T: NEXT K
5050 READ NUM: IF NUM < > CHECK THEN
     PRINT "TYPO IN LINE "LINE
    ::OK = 0
5060 LINE = LINE + 10: NEXT J
5070 IF OK = 0 THEN STOP
5080 PRINT  : INPUT "DID YOU REMEMBE
    R THE POKES? ";A$
5090 IF LEFT$ (A$,1) = "Y" THEN 5130

5100 PRINT : PRINT "SAVE THIS PROGRA
    M AND THEN"
5110 PRINT : INVERSE : PRINT "NEW:PO
    KE104,10:POKE2560,0:NEW": 
    NORMAL : PRINT
5120 PRINT "THEN RELOAD AND RUN.": S
    TOP
5130 RESTORE : FOR J = BOT TO HIGH S
```

```
    TEP 16
5140 FOR K = J TO J + 15: READ IT: P
    OKE (K),IT: NEXT K: READ I
    T: NEXT J
5150 PRINT : PRINT "BLOAD OKAY": CAL
    L BOT + 12: RETURN
5970 REM
5980 REM  MACHINE CODE DATA
5990 REM
6000 DATA 169,10,133,104,169,0,10,32
    ,75,214,169,76,141,245,168
    8
6010 DATA 3,169,31,141,246,3,169,8,1
    41,247,3,96,201,171,240,25
    ,1894
6020 DATA 201,176,240,36,201,190,240
    ,106,201,180,208,3,76,181,
    8,201,2448
6030 DATA 177,208,3,76,230,8,76,201,
    222,32,66,8,76,65,217,32,1
    697
6040 DATA 177,0,32,123,221,76,82,231
    ,169,3,32,214,211,165,185,
    72,1993
6050 DATA 165,184,72,165,118,72,165,
    117,72,169,176,72,32,66,8,
    32,1685
6060 DATA 183,0,201,0,240,38,201,58,
    240,34,201,44,240,30,32,16
    6,1908
6070 DATA 8,196,108,48,6,208,7,197,1
    07,16,3,76,150,225,32,249,
    1636
6080 DATA 234,32,106,221,160,0,32,11
    9,9,32,52,9,32,63,8,76,118
    5
6090 DATA 210,215,32,163,8,160,2,32,
    119,9,32,87,9,76,149,217,1
    520
6100 DATA 32,177,0,201,33,208,143,32
    ,177,0,32,227,223,133,0,13
    2,1750
6110 DATA 1,96,32,177,0,32,248,230,7
    2,201,176,240,13,201,171,2
    40,2130
6120 DATA 9,201,175,208,224,104,32,1
    77,0,72,198,161,208,4,104,
    76,1953
6130 DATA 31,8,32,177,0,32,227,223,3
    2,183,0,201,44,240,235,104
    ,1769
6140 DATA 104,104,96,32,163,8,141,16
    2,9,140,163,9,165,184,72,1
    65,1717
6150 DATA 185,72,160,2,32,129,9,32,5
    2,9,173,162,9,133,0,173,13
    32
6160 DATA 163,9,133,1,160,0,32,129,9
    ,32,87,9,104,133,185,104,1
    290
6170 DATA 133,184,32,177,0,76,107,21
    7,165,107,197,2,208,1,96,1
    33,1835
6180 DATA 2,169,7,208,2,169,5,24,101
    ,0,133,0,2,144,230,1,1197
6190 DATA 96,32,139,9,32,123,221,32,
    106,221,165,18,240,3,76,19
    8,1711
6200 DATA 8,32,27,9,166,0,164,1,32,4
    3,235,32,149,9,32,177,1116
6210 DATA 0,76,55,9,32,139,9,32,227,
    223,32,27,9,165,0,164,1199
6220 DATA 1,32,249,234,166,131,164,1
    32,32,43,235,32,149,9,32,1
    77,1818
6230 DATA 0,76,90,9,165,184,145,0,20
    0,165,185,145,0,96,177,0,1
    637
6240 DATA 133,184,200,177,0,133,185,
    96,165,107,133,2,32,40,9,7
    6,1672
6250 DATA 190,222,32,40,9,32,183,0,2
    01,33,208,2,104,104,96,0,1
    456
6260 DATA 104,104,96,0,0,0,0,0,0,0,0
    ,0,0,0,0,0,304
```

©



MEMOREX
FLEXIBLE DISCS

WE WILL NOT BE UNDERSOLD!! Call Free **(800)235-4137** for prices and information. Dealer inquiries invited and C.O.D.'s accepted.

PACIFIC EXCHANGES
100 Foothill Blvd.
San Luis Obispo, CA
93401. In Cal. call
(800)592-5935 or
(805)543-1037

VISA

TOLL FREE
Subscription
Order Line
800-345-8112
In PA 800-662-2444

This is all you really need!

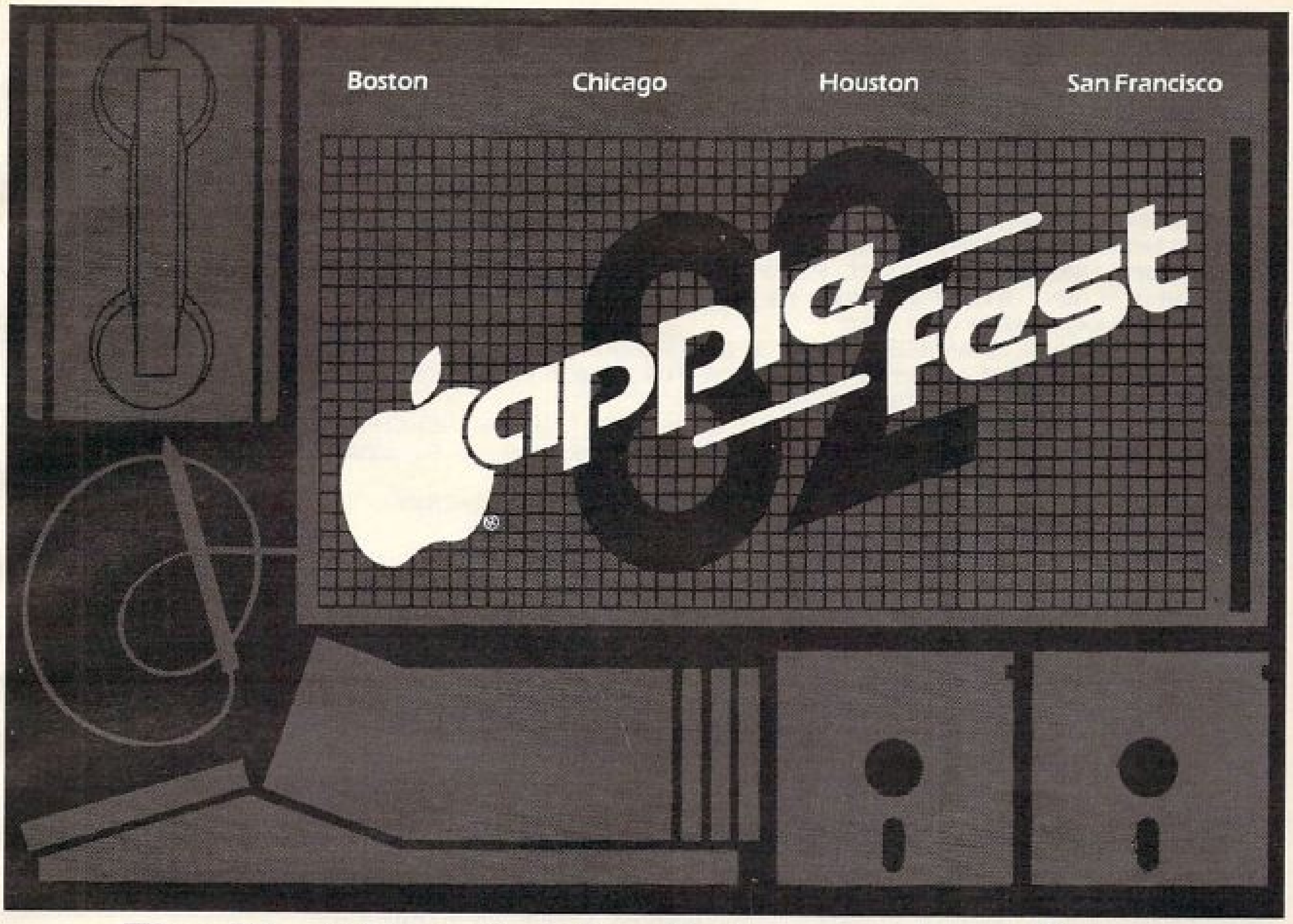
An Apple II®, a printer, a video screen, a modem, a disk drive, CPS Multifunction, and RAMPLUS+.

The CPS Multifunction card and cables are all you need to connect almost any printer, modem, or terminal to your Apple II®. It's really three cards in one. It provides the capabilities of a serial interface, parallel output interface, and real-time clock/calendar. It saves you money, power, and slots. The CPS Multifunction is compatible with DOS, CPM, PASCAL, and BASIC. Suggested retail price $239, including software. Cables $24.95 each.

The RAMPLUS+ card allows you to expand the available memory of your Apple II® to 80K. RAMPLUS+ has two 16K banks of RAM. The card is supplied with 16K of installed RAM, with an additional 16K of plug-in RAM available. It is compatible with DOS, CPM, PASCAL, and BASIC, also. Suggested retail price $189. Additional 16K RAM $24.95.

Mountain™ Computer
INCORPORATED
300 El Pueblo, Scotts Valley, CA 95066
TWX 910 598 4504   (408) 438-6650

The modem is a product of Hayes Microcomputer Products, Inc.
The video screen is a product of NEC.
The printer is a product of Epson America.
The Apple II® is a product of Apple Computer, Inc.
The disk drive is a product of Apple Computer, Inc.



The Most Spectacular Extravaganza Ever... For Apple Users

At Applefest '82 hundreds of manufacturers, distributors and dealers will showcase the entire spectrum of Apple-compatible products including computers, components, peripherals, plug-in cards, publications, gifts, magazines, services, accessories and software for home, office and school.

Hands-on centers and multimedia presentations will demonstrate the newest applications for business, education and entertainment.

Seminars and workshops, conducted by the world's leading Apple authorities, will detail new uses to make your Apple more enjoyable and more useful than you ever imagined.

You'll meet thousands of other Apple owners and find the newest of everything for your Apple under one roof . . . and for sale at super show prices.

So if you use an Apple . . . or are thinking about buying one, you won't want to miss a minute of Applefest '82.

Ticket & Hotel Information

Send your check and a note indicating the specific show you wish to attend. Tickets and hotel information will be mailed back to you. Tickets can also be purchased at the show. Make all checks payable to Northeast Expositions Inc. 824 Boylston Street, Chestnut Hill, Mass. 02167 Tel: 617 739 2000.

Applefest/Boston
Fri-Sun May 14-16, 1982
Hynes Auditorium
Show Hours: 11AM to 6PM Daily
Admission: $6 per day or $10 for 2 days, $15 for 3 days

Applefest/Chicago
Fri-Sun Nov 5-7, 1982
Arlington Park Racetrack/Exposition Center
Arlington Heights, Ill.
Show Hours: 1PM to 10PM Daily
Admission: $5 per day or $8 for 2 days, $12 for 3 days

Applefest/Houston
Fri-Sun Nov 19-21, 1982
Albert Thomas Convention Center
Show Hours: 1PM to 10PM Daily
Admission: $5 per day or $8 for 2 days, $12 for 3 days

Applefest/San Francisco
Fri-Sun Dec 3-5, 1982
Moscone Center
Show Hours: 1PM to 10PM Daily
Admission: $5 per day or $8 for 2 days, $12 for 3 days

Comment Your Catalog

Richard Cornelius
Department of Chemistry
Wichita State University
Wichita, KS

Since the first day that I had my Apple II, I have been frustrated by the inability to fully identify stored programs and files except by using long names. Wouldn't it be nice, for example, to have the date of the latest revision of a program stored along with its name? Of course, a person can always make the date part of the name, but I thought that there ought to be a better way. There is a better way. I have written a program to make writing comments in the catalog easy.

Control Characters

You may have already discovered that some control characters can be part of program and file names in the catalog. For example, a CTRL-J at the end of a program name is helpful in formatting the catalog. The CTRL-J is a linefeed which, when entered as the last character in a program name, has the effect of leaving an empty line between that program name and the next one when the catalog is listed. Another control character which can be inserted into a program name is CTRL-G which will make the Apple beep when the name of the program is listed in the catalog.

Most of the other control characters can be entered into program names, but generally they are not particularly useful. One application they do have is based on the fact that control characters in a name do not actually appear on the screen in the catalog, but they must be used in order to access the program on the disk. Their invisibility can provide a measure of security by preventing someone else from readily loading programs off of your disk. (See your Apple DOS manual for a program to detect most of these control characters.)

The control character that I have found useful in creating comments for the catalog is CTRL-H, the backspace character. This character cannot easily be entered directly into a program name. Typing CTRL-H is the same as pressing the left arrow; you can backspace over characters, but the character that you backspace over is deleted from the name as you backspace. The solution to this difficulty is to put CHR$(8) into a string variable that you use as the program name. In *immediate mode,* [not in a program – just type it on the screen directly] try going through the routine below using an initialized disk with only the HELLO program on it:

```
]CATALOG
DISK VOLUME 254
 A 002 HELLO
]D$=CHR$(4)
]NAME$="ABC"+CHR$(8)+CHR$(8)+"DEF"
]?D$"SAVE";NAME$
]CATALOG
DISK VOLUME 254
 A 002 HELLO
 A 004 ADEF
]LOAD ADEF
FILE NOT FOUND
]
```

The lines that start with a "]" prompt are the ones that I typed into the Apple. The others are those that the computer wrote. When I try to load ADEF the computer tells me FILE NOT FOUND because the name is not ADEF, but "ABC"+CHR$(8)+CHR$(8)+"DEF". Although the program name in the catalog appears to be four characters long, if you were to ask ?LEN(NAME$) you would find that it is actually eight characters long.

This information about CHR$(8) is really all that you need in order to be able to write comments into your catalog. You simply create a string variable that contains enough backspace characters to backspace over the letter that identifies the file type and the number that gives how many sectors are occupied on the disk by the file. Once all of that information is backspaced over, the desired comment is entered into the string. The string variable is then used as shown above to SAVE a program – any program. The "comment" is actually the name of a program – whatever program you had in memory when you do the SAVEing – but it doesn't look like a program name because the file type and sector-count information is missing.

Some Limitations

This commenting technique does have its limitations. Names of programs are limited to 30 characters by DOS. Since the first character of a name cannot be a control character, seven backspaces are needed to erase the information that is normally printed. The first character, plus these backspaces, consume eight of the available 30 characters, so only 22 characters can go into a comment. In addition, you have only limited control over where in the catalog the comment appears. This kind of comment is best used for disks on which people are not going to be making many changes. As long as you start with a fresh disk and put the files, programs, and comments onto the disk in the order you wish them to appear, the catalog will come out fine. If you modify programs in such a way as to change their length, then the order of items in the catalog may be changed and the comments will no longer be adjacent to the program name. One more limitation is that hard copies of the catalog are harder to make appear as nice as the screen listing of the commented catalog. If you try to print the catalog directly, the printer will backspace and overstrike the original characters.

This difficulty can be overcome by listing the catalog on the screen and then, using a program such as that by Jeff Schmoyer (**COMPUTE!** #6) to route the screen image to the printer. In spite of these limitations, I have prepared commented catalogs such as the one in Figure 1. Each line of letters is actually a program name, but the only programs of interest are the ones that have the file type and sector count next to them. The other program names serve only as comments, and the actual programs could be anything (or nothing).

Clearly typing all of these names with the CHR$(8) feature inserted could be quite a chore at the keyboard, so I wrote a program to enter the comments into the catalog. The program is called simply "Catalog Commenter" and is a short BASIC (Applesoft) program. The program shows just how long the name can be and lets you either erase or write names. It then gets a catalog so that you can see what you have done. Hitting any key clears the screen and takes you back to the beginning of the program. This program is the one that was used to prepare the catalog Figure 1. After the backspace characters, two spaces are inserted into the initial part of the string variable used for the name. This spacing makes the comments appear lined up with the sector count of the "real" program names in the catalog, but further limits the length of the comments to 20 characters.

Figure 1.

```
DISK VOLUME 254

 A 025 PH PLOT-BUFFER CAPACITY
   (MAIN PROGRAM WHICH
   LOADS OTHER FILES)

*B 002 OR LOADER & LINE ERASE
   (OVERLAYS HIRES PAGE
   2 ONTO PAGE 1 AND
   ERASES HIRES TEXT
   LINES. A$300; A$325)

*B 027 MZCHAR3
   (SPECIAL WHITE CHAR-
   ACTER SET. A$6000)

*B 006 INSTRUCTIONS
   (BINARY TEXT FILE OF
   INSTRUCTIONS.A$8000)

*B 034 COVER PAGE
   (BINARY HIRES FILE.
```

```
100 REM  ** CATALOG COMMENTER**
110 REM BY RICHARD CORNELIUS
120 REM CHEMISTRY DEPARTMENT
130 REM WICHITA STATE UNIV.
140 REM WICHITA, KS  67208
150 REM (316) 689-3120
160 REM  **INITIALIZATION**
170 D$= CHR$(13) + CHR$(4)
180 REM D$ SIGNALS DOS COMMAND
190 N$= CHR$(8) + CHR$(8) + CHR$(8)

200 REM CHR$(8) IS BACKSPACE
210 N$="A"+N$+N$+CHR$(8)+"  "
220 HOME: VTAB 5
230 REM  **GET COMMENT**
240 PRINT "TYPE IN COMMENT"
250 PRINT"---UP TO THIS LONG--"
260 INPUT"";C$
270 PRINT
280 PRINT"WRITE(W), ERASE(E), OR QU
    IT(Q)?";
290 GET G$
300 IF G$= "Q" THEN 410
310 IF G$ <> "E" AND G$ <>"W" THEN
    GOTO 220
320 REM  **CREATE PROGRAM NAME**
330 N$= N$ + C$
340 REM  **WRITE TO DISK**
350 IF G$= "E" THEN 370
360 PRINT D$"SAVE";N$:GOTO 380
370 PRINT D$"DELETE";N$
380 PRINT D$"CATALOG"
390 GET G$
400 IF G$ <> "Q" THEN 220
410 PRINT:PRINT"THE END"
```
©

Apple Addresses

Bill Grimm
Mountain View, CA

The Apple II uses three types of addressing depending upon the language being used. Apple's machine language uses hexadecimal addresses in the range from $0000 to $FFFF. Its Floating Point BASIC language uses decimal addresses in the range from 0 to 65535. Its Integer BASIC uses decimal addresses in the range from 0 to 32767 to -32767 to -1. This means that, if you want to address a particular memory location, you must choose the correct address for the language you are using. Since I program in all three languages and my references are a mixture from all three, I needed an address cross-reference program. So I wrote "Apple Addresses."

"Apple Addresses" can be used "as is" to convert one language's address to another's, and to give the high and low byte values which need to be poked into a BASIC program to store that address. Alternatively, you could extract the subroutines in Apple Addresses which convert between hex and decimal numbers and insert them in your own program. See the last paragraph of this article for more details.

The program begins by asking the user which of the six possible conversions he would like to make. This is followed by a request to select the way the results of the conversions are to be displayed. There are four possible displays:

1. single conversions displayed on the monitor one at a time.

2. Single conversions printed out on a Silentype printer* one at a time.

3. a range of conversions displayed on the monitor.

4. a range of conversions printed out on a Silentype printer*.

*With slight program modifications other printers could be used.

Subroutines

"Apple Addresses" makes extensive use of subroutines. This helps in organizing the program as well as making it shorter and easier to debug. The controlling or EXECutive routine is called Apple Addresses – Exec. It starts on line 100 and goes to line 310. Since a picture is worth a thousand words, I made what I call a *balloon diagram* (Figure 1) to show how data flows through the program. These are the conventions I used to make the diagram;

1. Each balloon represents a subroutine. The name of the subroutine and the line numbers where it is located are placed in the balloon.

2. Data flows through a subroutine in the direction of the arrows on the outside of the balloon.

3. Data flows between subroutines in the direction of the arrows on the *strings*.

4. If conditions are placed on what data flows through a subroutine, these conditions are written in along the *strings*.

As an additional aid for understanding how the program works I have included the following variable descriptions list:

A( ) — each A(I) holds the decimal equivalent value of the Ith hexadecimal numeral in the hex number being created from a decimal number — appropriate numbers are then added to convert these to ASCII codes.

A$( ) — holds the characters represented by the ASCII codes in A( ).

CHOICE — holds the number of the conversion chosen — see lines 120 to 178.

DVL — holds the decimal value of the number being converted — may be either FP or INT decimal.

DVL$ — is the string equivalent of DVL and is used in the output routines.

FLAG — if flag = 1 then an invalid number was entered and the program returns to get a new number.

FRST — holds the FP Basic address equivalent of the lowest address in the selected range.

FRST$ — holds the smallest address chosen — this address is then processed and stored in FRST.

HVL$ — holds the hex number selected or the hex number resulting from the conversion — if no hex numbers are involved then it holds the converted decimal number.

LST — holds the FP Basic address equivalent of the largest address in the selected range.

LST$ — holds the largest address chosen — this address is then processed and stored in LST.

N — holds the decimal equivalent of each hex numeral in a hex number being converted to a decimal number.

PHI% — holds the number that would be poked into the high byte when placing the address into memory.

PLO% — holds the number that would be poked into the low byte when placing the address into memory.

POK — holds the address from which PLO% and PHI% are derived.

SELECT — holds the type of output selected — see lines 462 to 470.

STP — holds the positive decimal stepping interval chosen.

STP$ — holds the stepping interval chosen which is later changed and stored in STP.

TB — the horizontal tab value desired.

TN — holds the intermediate numbers of the decimal address that is being converted into a hex address.

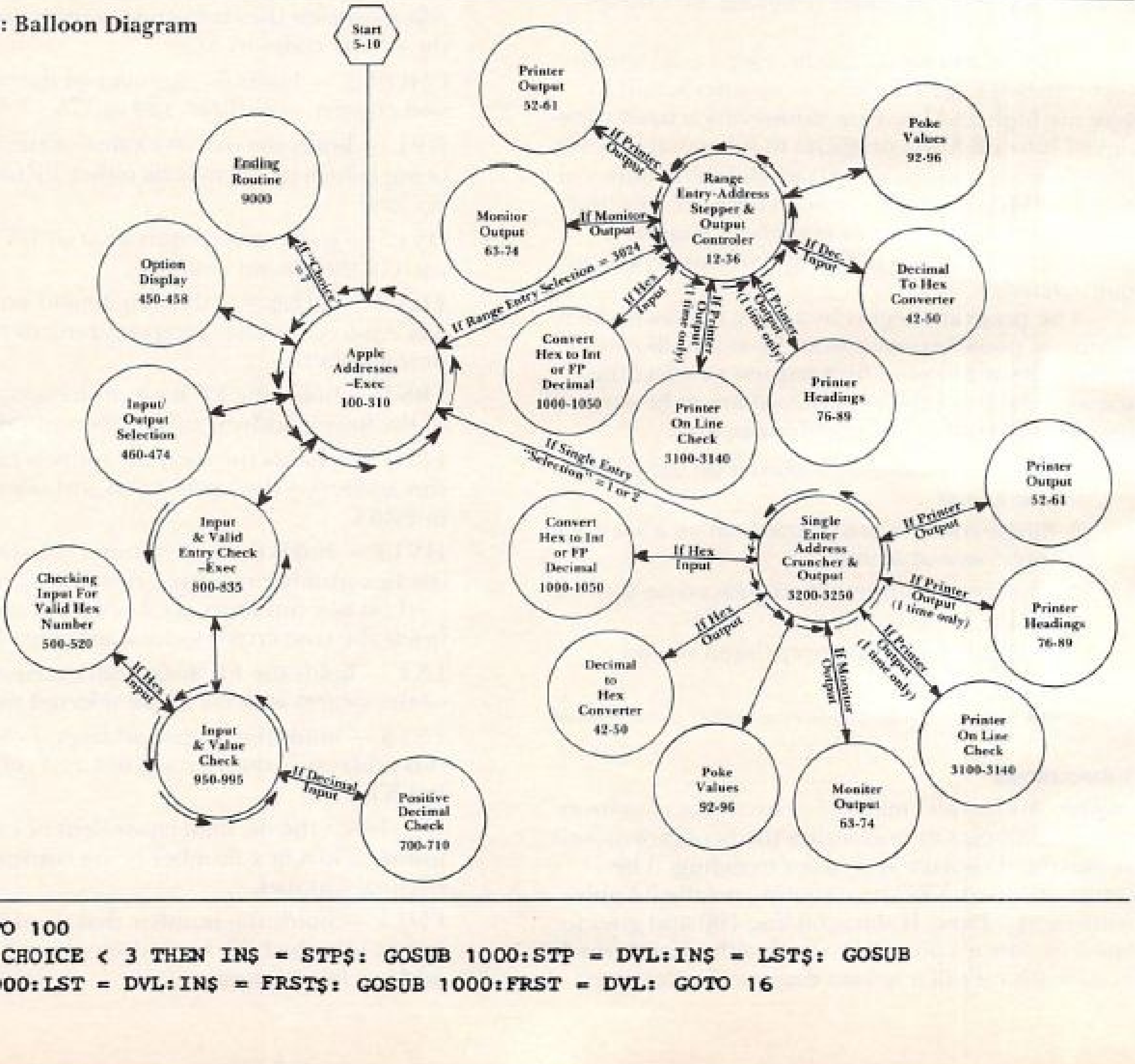
VTB — used to control the vertical tabbing of the monitor output.

Some Suggestions

I have found that the easiest way to debug a program while I am entering it is to first type in the EXEC program. Then, if I place return statements at all the branching locations, I can check the EXEC for bugs. Once the EXEC is free of bugs, I add one subroutine at a time in the order that the EXEC uses them, checking for bugs as I go.

If you have a need for subroutines which convert numbers from hex to decimal or from decimal to hex, two subroutines in this program may be of help. The first is called "decimal to hex converter" (lines 42 to 50). The input to this routine is TN which must hold a positive decimal number <65536. The output is HVL$ which holds the hex equivalent to the number in TN. The second is called "convert hex to INT or FP decimal" (lines 1000 to 1050). The input to this routine is HVL$ which must hold a hex number <=$FFFF and choice. If choice = 1 then you get the positive decimal equivalent. Otherwise you get Int BASIC's equivalent. The output is a decimal number in DVL.

Figure 1: Balloon Diagram


Start 5-10
Ending Routine 9000
If "Choice" = 7
Option Display 450-458
Apple Addresses -Exec 100-310
Input/ Output Selection 460-474
Input & Valid Entry Check -Exec 800-835
Checking Input For Valid Hex Number 500-520
If Hex Input
Input & Value Check 950-995
If Decimal Input
Positive Decimal Check 700-710
If Range Entry Selection = 3024
Printer Output 52-61
If Printer Output
Monitor Output 63-74
If Monitor Output
Range Entry-Address Stepper & Output Controler 12-36
Poke Values 92-96
If Dec. Input
Decimal To Hex Converter 42-50
If Hex Input
Convert Hex to Int or FP Decimal 1000-1050
If Printer Output (1 time only)
Printer On Line Check 3100-3140
If Printer Output (1 time only)
Printer Headings 76-89
If Single Entry "Selection" = 1 or 2
Convert Hex to Int or FP Decimal 1000-1050
If Hex Input
Single Enter Address Cruncher & Output 3200-3250
If Printer Output
Printer Output 52-61
If Printer Output (1 time only)
Printer Headings 76-89
If Hex Output
Decimal to Hex Converter 42-50
If Monitor Output
If Printer Output (1 time only)
Poke Values 92-96
Moniter Output 63-74
Printer On Line Check 3100-3140

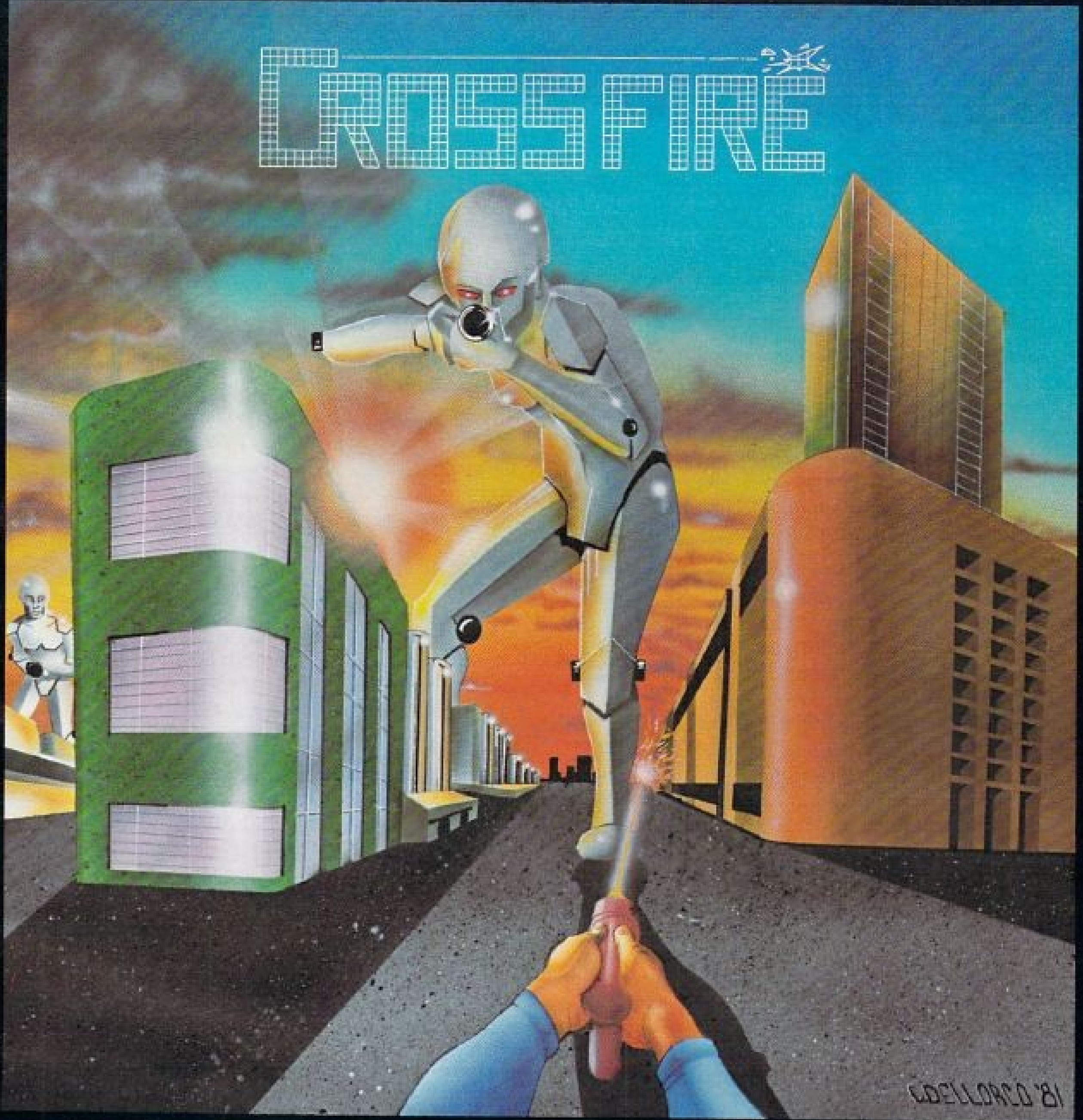
```
10  GOTO 100
12  IF CHOICE < 3 THEN IN$ = STP$: GOSUB 1000:STP = DVL:IN$ = LST$: GOSUB
     1000:LST = DVL:IN$ = FRST$: GOSUB 1000:FRST = DVL: GOTO 16
```

```
14 STP =  VAL (STP$):LST =  VAL (LST$):FRST =  VAL (FRST$)
16 VTB = 7:TB = 1: IF SELECT = 4 THEN  GOSUB 3100: POKE  - 12526,83: PR#
    1: PRINT : PRINT "CONVERTING FROM ";: ON CHOICE GOSUB 76,78,80,82,84
    ,86: POKE  - 12526,80
18  IF LST < 0 THEN LST = LST + 65536: IF FRST < 0 THEN FRST = FRST + 655
    36
19  FOR DVL = FRST TO LST STEP STP: IF CHOICE <  > 4 OR CHOICE <  > 6 THEN
    TN = DVL: GOSUB 42
20  IF CHOICE = 3 AND DVL > 32767 OR CHOICE = 4 AND DVL > 32767 OR CHOICE
    = 2 AND DVL > 32767 OR CHOICE = 6 AND DVL > 32767 THEN DVL = DVL -
    65536
22  IF CHOICE = 4 THEN HVL$ =  STR$ (DVL): IF DVL < 0 THEN HVL$ =  STR$ (
    DVL + 65536)
24  IF CHOICE = 6 THEN HVL$ =  STR$ (DVL): IF DVL < 0 THEN DVL = DVL + 65
    536
26  GOSUB 92
28  IF SELECT = 4 THEN  GOSUB 52: GOTO 32
30  GOSUB 62
32  IF DVL < 0 THEN DVL = DVL + 65536
34  NEXT DVL: IF SELECT = 4 THEN  PRINT : PR# 0
36  RETURN
42 HVL$ = "": FOR I = 4 TO 1 STEP  - 1:A(5 - I) =  INT (TN / (16 ^ (I - 1
    ))):TN = TN - (A(5 - I) * (16 ^ (I - 1))): NEXT I
44  FOR I = 1 TO 4: IF A(I) < 10 THEN A(I) = A(I) + 48: GOTO 48
46 A(I) = A(I) + 55
48 A$(I) =  CHR$ (A(I)):HVL$ = HVL$ + A$(I): NEXT I
50  RETURN
52 DVL$ =  STR$ (DVL): IF CHOICE < 3 THEN 58
54  PRINT  SPC( 6 -  LEN (DVL$));DVL$;: IF CHOICE = 5 OR CHOICE = 3 THEN
     PRINT ">$";HVL$; SPC( 1);: GOTO 59
56  PRINT ">"; SPC( 6 -  LEN (HVL$));HVL$;: GOTO 59
58  PRINT " $"; SPC( 4 -  LEN (HVL$));HVL$;">"; SPC( 6 -  LEN (DVL$));DVL
    $;
59  PRINT  SPC( 9 -  LEN (PLO$));PLO$; SPC( 14 -  LEN (PHI$));PHI$;:TB =
    TB + 39: IF TB > 42 OR SELECT = 2 THEN TB = 1: PRINT
60  HTAB TB: IF TB = 40 THEN  PRINT  SPC( 3);
61  RETURN
62  REM
63 DVL$ =  STR$ (DVL): VTAB VTB: HTAB TB: IF CHOICE < 3 THEN 68
64  PRINT  SPC( 6 -  LEN (DVL$));DVL$;: IF CHOICE = 5 OR CHOICE = 3 THEN
     PRINT ">$";HVL$; SPC( 2);: GOTO 70
66  PRINT ">"; SPC( 6 -  LEN (HVL$));HVL$; SPC( 1);: GOTO 70
68  PRINT "$0000>";: HTAB TB + 5 -  LEN (HVL$): PRINT HVL$;: HTAB TB + 12
    -  LEN (DVL$): PRINT DVL$; SPC( 2);
70  PRINT  SPC( 8 -  LEN (PLO$));PLO$; SPC( 14 -  LEN (PHI$));PHI$:VTB =
    VTB + 1: IF VTB > 23 THEN  HTAB 3: INPUT "PRESS <RETURN> TO CLEAR SC
    REEN";IN$: HOME :VTB = 6:TB = 1: GOTO 72
71  GOTO 74
72  IF IN$ = "Q" THEN  POP : GOTO 100
73  IF SELECT = 3 THEN VTB = 7
74  RETURN
76  PRINT "HEX TO FP DECIMAL": GOSUB 88: RETURN
78  PRINT "HEX TO INT DECIMAL": GOSUB 88: RETURN
80  PRINT "INT DECIMAL TO HEX": GOSUB 88: RETURN
82  PRINT "INT DECIMAL TO FP DECIMAL": GOSUB 88: RETURN
84  PRINT "FP DECIMAL TO HEX": GOSUB 88: RETURN
86  PRINT "FP DECIMAL TO INT DECIMAL": GOSUB 88: RETURN
88  IF SELECT = 2 THEN  PRINT : PRINT " CONVERSION   POKE LO BYTE   POKE H
    I BYTE": RETURN
```

```
89  PRINT : PRINT " CONVERSION   POKE LO BYTE  POKE HI BYTE   CONVERSION
      POKE LO BYTE  POKE HI BYTE": RETURN
92 POK = DVL: IF POK < 0 THEN POK = POK + 65536
94 PHI% = POK / 256:PLO% = POK - PHI% * 256
96 PHI$ =  STR$ (PHI%):PLO$ =  STR$ (PLO%): RETURN
100  POKE  - 16298,0: TEXT : HOME :FLAG = 0
110  VTAB 7
120  PRINT " 1. CONVERT HEX ADDRESSES TO FP BASIC": PRINT
130  PRINT " 2. CONVERT HEX ADDRESSES TO INT BASIC": PRINT
135  PRINT " 3. CONVERT INT BASIC ADDRESSES TO HEX": PRINT
140  PRINT " 4. CONVERT INT BASIC ADDRESSES TO FP": PRINT
150  PRINT " 5. CONVERT FP BASIC ADDRESSES TO HEX": PRINT
160  PRINT " 6. CONVERT FP BASIC ADDRESSES TO INT": PRINT
162  PRINT " 7. QUIT": PRINT
165  PRINT : PRINT "NOTE: ENTERING A 'Q' AT ANY POINT            RETURNS
     YOU TO THIS MENU."
170  VTAB 4: INPUT "CHOOSE ONE:";IN$
175  IF IN$ = "7" THEN 9000
178 CHOICE =  VAL (IN$): IF CHOICE < 1 OR CHOICE > 6 THEN 100
180  GOSUB 450: GOSUB 460: HOME : VTAB 1: HTAB 13: ON SELECT GOTO 190,195
     ,200,210
190  PRINT ": SINGLE ENTRY : MONITOR": GOTO 220
195  PRINT ": SINGLE ENTRY : PRINTER": GOTO 220
200  PRINT ": RANGE ENTRY : MONITOR": GOTO 220
210  PRINT ": RANGE ENTRY : PRINTER"
220  HOME : IF SELECT < 3 THEN  PRINT "ENTER NUMBER": GOTO 250
230  PRINT "FIRST NUMBER";: HTAB 22: PRINT "LAST NUMBER"
240  PRINT "STEPPING INTERVAL"
250  FOR I = 0 TO 39: PRINT  CHR$ (45);: NEXT I: PRINT " CONVERSION   POK
     E LO BYTE  POKE HI BYTE": POKE 34,6: IF SELECT < 3 THEN  POKE 34,5
260  HOME
280 CNT = 0:TB = 1:VTB = 7: IF SELECT < 3 THEN VTB = 6
290  GOSUB 800
300  ON SELECT GOSUB 3200,3200,12,12: IF SELECT < 3 THEN 290
310  VTAB 24: HTAB 5: CALL  - 868: INPUT "PRESS <RETURN> TO CONTINUE.";IN
     $: GOTO 100
450  HOME : HTAB 4: ON CHOICE GOSUB 452,456,458,455,454,457: FOR I = 0 TO
     39: PRINT  CHR$ (45);: NEXT I: POKE 34,2: RETURN
452  PRINT "HEX->FP": RETURN
454  PRINT "FP->HEX": RETURN
455  PRINT "INT->FP": RETURN
456  PRINT "HEX->INT ": RETURN
457  PRINT "FP->INT": RETURN
458  PRINT "INT->HEX": RETURN
460  HOME : VTAB 8
462  PRINT "  1. SINGLE ENTRY - MONITOR OUTPUT": PRINT
463  PRINT "  2. SINGLE ENTRY - PRINTER OUTPUT": PRINT
464  PRINT "  3. RANGE  ENTRY - MONITOR OUTPUT": PRINT
466  PRINT "  4. RANGE  ENTRY - PRINTER OUTPUT": PRINT
468  VTAB 6: INPUT "CHOOSE ONE:";IN$: IF IN$ = "Q" THEN  POP : GOTO 100
470 SELECT =  VAL (IN$)
472  IF SELECT < 1 OR SELECT > 4 THEN 460
474  RETURN
500  FOR I = 1 TO  LEN (IN$): IF  ASC ( MID$ (IN$,I,1)) > 70 OR  ASC ( MID$
     (IN$,I,1)) < 48 THEN 520
510  IF  ASC ( MID$ (IN$,I,1)) > 57 AND  ASC ( MID$ (IN$,I,1)) < 65 THEN  520
512  NEXT I: RETURN
520 FLAG = 1: RETURN
700  FOR I = 1 TO  LEN (IN$)
```

```
705  IF  ASC ( MID$ (IN$,I)) > 57 OR  ASC ( MID$ (IN$,I)) < 48 THEN 710
709  NEXT I: RETURN
710 FLAG = 1: RETURN
800  IF SELECT > 2 THEN 815
805  VTAB 3: HTAB 13: CALL  - 868: GOSUB 950: IF FLAG = 1 THEN FLAG = 0: GOTO
     805
810  GOTO 835
815  VTAB 3: HTAB 13: POKE 33,21: CALL  - 868: GOSUB 950:FRST$ = IN$: POKE
     33,40: IF FLAG = 1 THEN FLAG = 0: GOTO 815
820  VTAB 3: HTAB 33: CALL  - 868: GOSUB 950:LST$ = IN$: IF FLAG = 1 THEN
     FLAG = 0: GOTO 820
825  VTAB 4: HTAB 18: CALL  - 868: GOSUB 950:STP$ = IN$: IF DVL < 0 THEN
     FLAG = 1
830  IF FLAG = 1 THEN FLAG = 0: GOTO 825
835  RETURN
950  IF CHOICE > 2 THEN 970
955  INPUT "=$";IN$: IF IN$ = "Q" THEN  POP : POP : GOTO 100
957  IF IN$ = "" THEN FLAG = 1: GOTO 995
960  IF  LEN (IN$) > 4 THEN FLAG = 1: GOTO 995
965  GOSUB 500: GOTO 995
970  INPUT "=";IN$: IF IN$ = "Q" THEN  POP : POP : GOTO 100
972  IF IN$ = "" THEN FLAG = 1: GOTO 995
975  IF CHOICE < 5 AND  VAL (IN$) <  - 32767 THEN FLAG = 1: GOTO 995
977  IF CHOICE < 5 AND  VAL (IN$) > 32767 THEN FLAG = 1: GOTO 995
980  IF CHOICE > 4 AND  VAL (IN$) < 0 THEN FLAG = 1: GOTO 995
983  IF CHOICE > 4 AND  VAL (IN$) > 65535 THEN FLAG = 1: GOTO 995
985 DVL =  VAL (IN$): IF DVL < 0 THEN IN$ =  MID$ (IN$,2): GOSUB 700:IN$ =
      STR$ (DVL + 65536): GOTO 995
990  GOSUB 700
995  RETURN
1000 HVL$ = IN$
1010 DVL = 0: FOR I = 1 TO  LEN (IN$): IF  ASC ( MID$ (IN$,I,1)) > 64 THEN
     N =  ASC ( MID$ (IN$,I,1)) - 55
1018  IF  ASC ( MID$ (IN$,I,1)) < 64 THEN N =  ASC ( MID$ (IN$,I,1)) - 48
1020 DVL = DVL + N * 16 © ( LEN (IN$) - I): NEXT I
1030  IF CHOICE = 1 THEN 1050
1040  IF DVL > 32767 THEN DVL = DVL - 65536
1050  RETURN
3100  FOR I = 1 TO 7
3110 J =  - 16384 + 256 * I
3120  IF  PEEK (J + 23) = 201 AND  PEEK (J + 55) = 207 AND  PEEK (J + 76)
      = 234 THEN  RETURN
3130  NEXT I
3140  HOME : VTAB 10: PRINT "NO SILENTYPE PRINTER INSTALLED.": PRINT "SEL
     ECTION ABORTED!": FOR K = 1 TO 3000: NEXT K: POP : RETURN
3200  IF CHOICE < 3 THEN  GOSUB 1000: GOSUB 92: GOSUB 62: GOTO 3230
3210  IF CHOICE = 3 OR CHOICE = 5 THEN TN =  VAL (IN$): GOSUB 42: GOSUB 9
     2: GOSUB 62: GOTO 3230
3220 HVL$ = IN$: IF CHOICE = 6 AND  VAL (IN$) > 32767 THEN HVL$ =  STR$ (
     DVL - 65536)
3225  GOSUB 92: GOSUB 62
3230  IF SELECT = 2 AND CNT = 0 THEN  GOSUB 3100: POKE  - 12526,83: PR# 1
     : PRINT : PRINT "CONVERTING FROM ";: ON CHOICE GOSUB 76,78,80,82,84,
     86:CNT = CNT + 1
3240  IF SELECT = 2 THEN  PR# 1: GOSUB 52: PR# 0
3250  RETURN
9000  POKE  - 16300,0: POKE  - 16298,0: TEXT : CALL  - 936: POKE  - 16368
     ,0: END
```

©



They have landed and are taking over the city. Steadily they are making their way across the city, destroying everything in their paths. The town has been evacuated and your regiment has retreated leaving you, alone in the city, at the mercy of the aliens.

The aliens have you surrounded, and laser shots fly from all directions. Your movements are confined but you haven't given up. If you're going to live, you'll have to concentrate on where the shots are coming from and where you're going because if you don't, you'll get caught in the **CROSS FIRE**.

CROSS FIRE is a unique new game by JAY SULLIVAN featuring HI-RES graphics and sound, smooth quick animation, and some of the best arcade challenge available anywhere. **CROSS FIRE** runs on any 48K APPLE II/II PLUS DOS 3.2 or 3.3 and is available now for $29.95 on disk from your local computer store or you may order directly from......

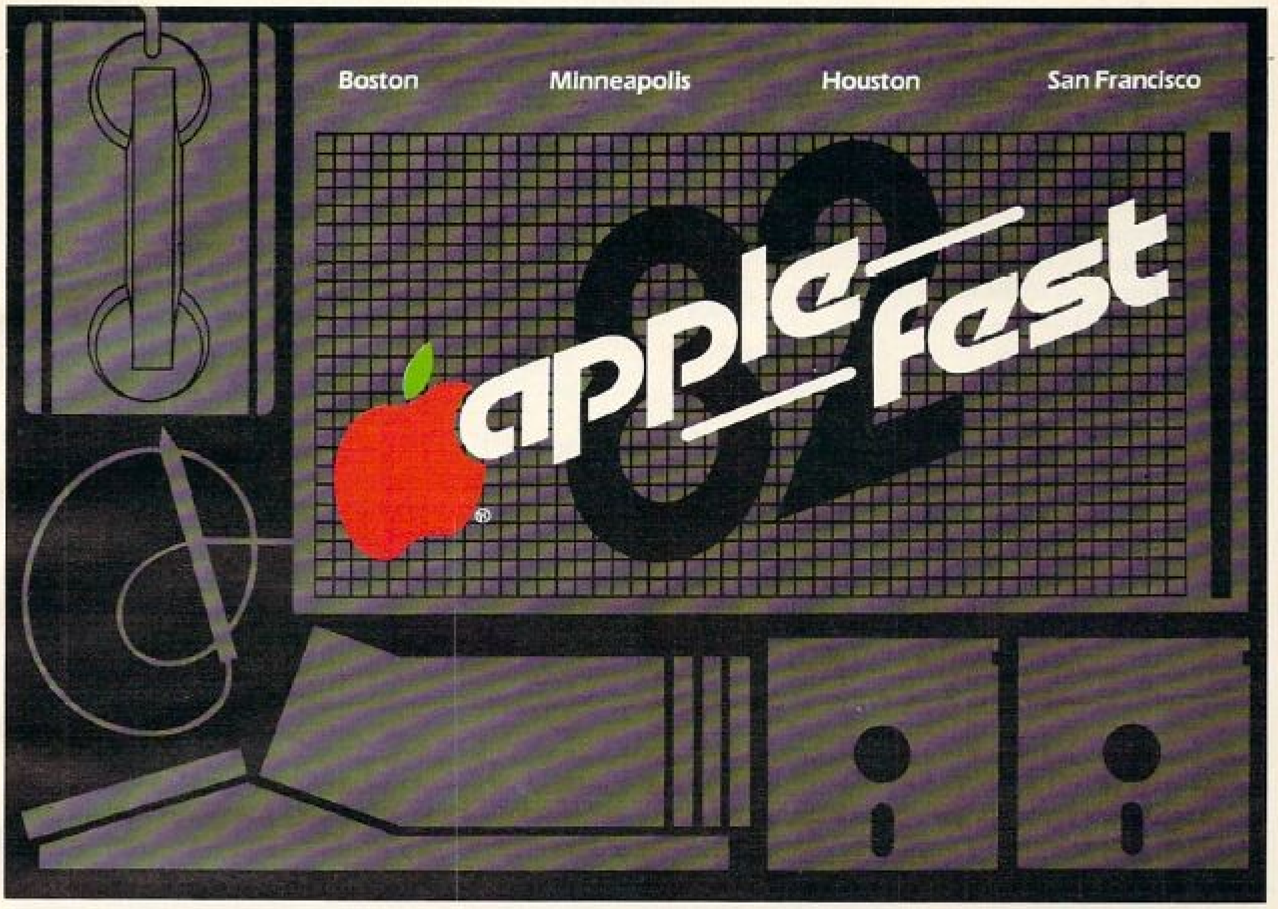
NOW AVAILABLE
FOR ATARI
PERSONAL COMPUTERS
WITH 32K AND DISK DRIVE

ON-LINE systems

36575 Mudge Ranch Road - Coarsegold, CA 93614 - 209-683-6858

ADD $1.00 FOR DIRECT ORDERS

VISA, MASTER CHARGE, C.O.D. or CHECKS ACCEPTED



The Most Spectacular Extravaganza Ever... For Apple Users

At Applefest '82 hundreds of manufacturers, distributors and dealers will showcase the entire spectrum of Apple-compatible products including computers, components, peripherals, plug-in cards, publications, gifts, magazines, services, accessories and software for home, office and school.

Hands-on centers and multimedia presentations will demonstrate the newest applications for business, education and entertainment.

Seminars and workshops, conducted by the world's leading Apple authorities, will detail new uses to make your Apple more enjoyable and more useful than you ever imagined.

You'll meet thousands of other Apple owners and find the newest of everything for your Apple under one roof . . . and for sale at super show prices.

So if you use an Apple . . . or are thinking about buying one, you won't want to miss a minute of Applefest '82.

Ticket & Hotel Information
Send your check and a note indicating the specific show you wish to attend. Tickets and hotel information will be mailed back to you. Tickets can also be purchased at the show. Make all checks payable to Northeast Expositions Inc. 824 Boylston Street, Chestnut Hill, Mass. 02167 Tel: 617 739 2000.

Exhibitor Information
For specific exhibitor information on one or all of the Applefest '82 shows call Northeast Expositions at the telephone number above.

Applefest/Boston
Fri-Sun May 14-16, 1982
Hynes Auditorium
Show Hours: 11AM to 6PM Daily
Admission: $6 per day or $10 for 2 days, $15 for 3 days

Applefest/Minneapolis
Thurs-Sun Sept 16-19, 1982
Minnesota Auditorium and Convention Hall
Show Hours: 11 AM to 6 PM Daily
Admission: $5 per day or $8 for 2 days, $12 for 3 days, $15 for 4 days

Applefest/Houston
Fri-Sun Nov 19-21, 1982
Albert Thomas Convention Center
Show Hours: 1PM to 10PM Daily
Admission: $5 per day or $8 for 2 days, $12 for 3 days

Applefest/San Francisco
Fri-Sun Dec 3-5, 1982
Moscone Center
Show Hours: 1PM to 10PM Daily
Admission: $5 per day or $8 for 2 days, $12 for 3 days

Applefest is produced by Northeast Expositions Inc. and is sanctioned by Apple Computer Inc. and The Boston Computer Society.

*Apple and Applefest are registered trade and service marks of Apple Computer Inc.

Customizing Apple's COPY Program

Roger B. Chaffee
Menlo Park, California

How many times have you used the COPY or COPYA program from the Apple DOS Master Disk? How many times have you had to tell the program that you wanted Slot 6, Drive 1, even though you have only one drive, and it's *always* in Slot 6, Drive 1? Every time, right?

Well, the nice people at Apple who brought you the COPY programs have made a system that works well, and is also easy to modify. Many programs, like Muffin and the DOS itself, are written in machine language (ML), and are very difficult to modify, or even to examine and understand. The main routines of COPY and COPYA, however, are written in BASIC, and only the nitty-gritty of buffer management and interface with the RWTS routine are done in ML.

Take a look at COPYA, which is written in AppleSoft BASIC, or at COPY if you want to use Integer BASIC. To look at the program, type LOAD COPYA [LOAD COPY] and then LIST. You can also list specific line numbers or ranges. Here's an explanation of some of the program lines. Later, we'll look at how to change them to fit your own system.

(Some of the information in this article depends on which program you are looking at, as in the LOAD commands above. When this is true, the information given is for the COPYA program, and the information for COPY follows it in brackets.)

Line 70 [90]: Load in the ML routines. (There is a hidden CTRL-D in this line, which will disappear if you use the BASIC editor to replace the line.)

Line 90 [120]: CALL 704 to initialize the ML routines. CS gets the current slot number, which is probably six, and location 720 gets the current drive number, either one or two. The values come from the IOB used by the DOS.

Lines 100,110 [130,140]: Set locations 715, 716 to the page numbers of the first and last pages that the ML routines will use for buffer space. They are different for AppleSoft and Integer because the two BASICs use memory and pointers differently.

Line 130 [150]: Call subroutines to ask the user for the slot and drive of the Master ("Original") diskette. MS gets the Master Slot number (zero to seven), and MD gets the Master Drive number (one or two).

Line 132 [160]: Call subroutines to ask the user for the slot and drive of the Slave ('Duplicate') diskette. SS gets the Slave Slot number (one to seven), and SD gets the Slave Drive number (one or two).

Line 165 [190]: Here's where the program stops asking for information, and starts copying the disk. Do not modify anything from here through the END statement in Line 305 [420] unless you are very sure of what you are doing! Before this line, MS and MD must be set to the slot and drive numbers for the Master diskette, and SS and SD must be set to the slot and drive numbers for the Slave diskette.

Line 310 [430]: This subroutine asks for a slot number, and is called for both Master and Slave slots. It also prints information on the screen, so you shouldn't simply replace it by a RETURN statement.

Line 320 [440]: This subroutine asks for a drive number, and is called for both Master and Slave slots. Again, don't just remove it, because it prints as well as asking you for the number.

Line 330 to 340 [450 to 460]: This subroutine gets a one-digit number, from L to H, from the keyboard. If RETURN is pressed, it uses the number already in N.

Line 350 [470]: This subroutine prints "DEFAULT=" and then puts on a flashing cursor.

Line 360 [480] to the end: This subroutine prompts you to change diskettes, if MS=SS and MD=SD. Otherwise, it just returns, with no action.

Okay, so that's how COPY gets it's parameters. But how can you change things so that it works just right for your own Apple system, which has a fixed number of drives in fixed slot locations?

Case 1: One Drive

The first case assumes you have only one drive, which is always in slot 6, drive 1. There is no choice. You always want slot 6, drive 1. The simplest way to make this happen is probably to replace lines 130 and 132 [150 and 160] by the statement

```
MS=6: SS=6: MD=1: SD=1
```

Sams offers five books designed to help you explore the basic workings of your Apple microcomputer. Here you'll find all of the information necessary to understand and use the limitless computing power of the Apple system. With these great books from Sams, your venture into the world of the Apple microcomputer will be a fruitful one!

APPLESOFT LANGUAGE Beginning programmers are given a comprehensive, step-by-step introduction to Applesoft, the language of the Apple II microcomputer. No. 21811. $10.95

INTIMATE INSTRUCTIONS IN INTEGER BASIC This complete introductory overview of Integer BASIC enables even the most nontechnical reader to write and use simple programs. No. 21812. $7.95

6502 SOFTWARE DESIGN The 6502 microprocessor is the heart of your Apple microcomputer. 6502 Software Design introduces you to programming in 6502 assembly language allowing you to perform high-speed processing and peripheral control not generally possible with higher-language programs. No. 21656. $12.95

MOSTLY BASIC: APPLICATIONS FOR YOUR APPLE® II Twenty-eight BASIC language programs for your Apple II microcomputer are thoroughly outlined and discussed. These programs are useful in all facets of the home and business. No. 27189. $12.95

APPLE® INTERFACING You are provided with real, tested circuits which interface to your Apple microcomputer for remote control, data acquisition, and control of external electronic and electromechanical devices. No. 21862. $10.95

Each book presents a different slice of the intricate workings of your Apple microcomputer. To take full advantage of this powerful machine, you must understand it from the inside out.

To order these Sams books or to get the name of your local Sams retailer, call toll free and reference this number: AD163

800-428-3696

SAMS BOOKS™
Howard W. Sams & Co., Inc.
4300 West 62nd Street, P.O. Box 7092
Indianapolis, IN 46206

GET TO THE CORE OF YOUR APPLE®

Mostly BASIC: Applications for Your APPLE
APPLE INTERFACING
6502 SOFTWARE DESIGN
INTIMATE INSTRUCTIONS IN INTEGER BASIC
APPLESOFT LANGUAGE

Offer good in USA only and expires 7/31/82. Prices subject to change without notice.

but that won't give you the nicely formatted screen to tell you what's happening. Here's a better way: replace the subroutine which gets the input values.

```
AppleSoft:     330 K=141
[Integer:      450 REM]
```

Then when any value is needed, the default will be used. To make it work just right, you also need to go back to line 132 [160] and replace the statement N=3-MD by the statement N=MD, which will set the slave drive to the same value (1) as the master drive.

Case 2: Two Drives The Easy Way

Suppose you have two drives, and you want the program to decide which gets the master and which gets the slave diskette. Make the same subroutine replacement as before:

```
AppleSoft:     330 K=141
[Integer:      450 REM]
```

Now whichever drive you used most recently will be the master, and the other will be the slave.

Case 3: Two Drive The Right Way

Finally, suppose you have two drives, both in the same slot, but you want to be able to choose which one will be the master. This time you can't just remove the subroutine which gets the numbers, because you want the program to ask you for the drive number. Instead, you can stop the program from asking you for the slot numbers:

```
AppleSoft:
  Change GOSUB 330 to GOSUB 340 in Line 310.
[Integer:
  Change GOSUB 450 to GOSUB 460 in Line 430.]
```

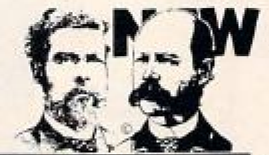
In this last case, to make sure that the slave drive is the "other" one, that is the one you didn't specify for the master, the simplest change is this:

```
AppleSoft:
  Move statement 330 to 331
  Insert
    330 IF LEFT$(I$,1)="D" THEN 340
[Integer:
  Move statement 450 to 451
  Insert
    450 IF I$(1,1)="D" THEN 460
```

Leave the rest of the line alone.

Apology And Exhortation

As you read this, it probably sounds more complicated than it really is. If you have done any BASIC programming on your Apple, you already know how to modify programs, and COPY is just another BASIC program. There's no reason that you should stick with a general-purpose program, when it is easy and maybe even instructive to fit the Apple's general program to your specific needs. ©



NOW® Apple Utility Disks!
With Free PEEKS & POKES Wall Chart!

utility city
21 of our most-asked-for Apple Utilities
by Bert Kersey 48K Applesoft ROM

For YOUR Big Apple—Our 21 most-asked-for Apple Utilities on one big disk—**List formatter** makes properly spaced & indented listings with printer page breaks. Each program statement is on a new line with If-Then's & Loops called out; a great de-bugger! **Catalog in any number of columns** & any page-width to CRT or printer. Automatically post the Run-Number & last-used Date in your programs. **Make any command invisible** in your listings; Access program lines in memory for garbage repair & "illegal" alteration; Quickly sort & store info on disk; Run any Applesoft file while another stays intact; Move lines within your programs; Renumber to 65535; Save inverse, INVISIBLE & trick file names; Convert dec to hex & binary, or INT to FP; Append programs; dump the text screen to ANY printer . . . More too: **21 Programs Total!**

PLUS . . . APPLE TIP BOOK NUMBER THREE!
40 PAGES of new tips, tricks & articles—"DOS Trickery," "Copy Stoppers," "Programming the Reset Key" & much more. Plus understandable explanations of how each U-City Program works—One of the best Apple learning tools on the market today!

$29.50
✓ **Utility City** on Applesoft Disk
✓ 40-page Beagle Bros Apple Tip Book #3
✓ Apple PEEKS, POKES & POINTERS Chart

alpha plot
Hi-Res Apple Graphics/Text Utility
by Bert Kersey & Jack Cassidy 48K REQUIRED

HI-RES DRAWING: Create hi-res pictures & charts **appendable to your programs.** Keyboard or Paddle control; Optional Xdraw Cursor (see lines before you draw!); Any color mix or REVERSE (opposite of background); Circles, Boxes & Ellipses, filled or not. Bonus Programs too—**SCRUNCHER** stores hi-res in as little as 1/3 normal disk space. **SHIFTER** transfers any portion of the hi-res screen. Also superimpose hi-res images and convert Hi-Res to Lo-Res & back for fascinating abstracts!

HI-RES TEXT: Beautiful upper & lower case with Descenders, color or reverse, Positionable anywhere (NOT restricted by Htabs & Vtabs). Professional looking **PROPORTIONAL SPACING!** Adjustable Type Size, Leading (line spacing) & Kerning (letter spacing). Multi-directional typing; up, down, even backwards!

PLUS . . . APPLE TIP BOOK NUMBER FOUR!
40 PAGES of tips for making the most of your Apple's advanced graphics capabilities. Alpha instructions plus shape table tips, new tilting & color tricks and fascinating animation experiments!

$39.50
✓ **Alpha Plot** on Applesoft Disk
✓ 40-page Beagle Bros Apple Tip Book #4
✓ Apple PEEKS, POKES & POINTERS Chart

Beagle Bros MICRO SOFTWARE
4315 Sierra Vista
San Diego,
CA 92103
714-296-6400

WITH EACH ORDER, YOU WILL RECEIVE . . .
☐ **Dos Boss, Utility City** or **Alpha Plot** on Applesoft Disk
☐ A 40-page **Apple Tip Book** (each entirely different)
☐ An 11x17 Apple **PEEKS, POKES & POINTERS** Chart

dos boss
DISK COMMAND EDITOR
by Bert Kersey & Jack Cassidy

A classic utility you will ENJOY! **Rename Commands:** "Catalog" can be "Cat," etc. **Save-Protect your programs;** Unauthorized copy attempts produce "Not Copyable" message. 1-Key Program Selection from catalog (extremely handy!). Catalog Customizer—Change Disk Volume message to your title; Omit or alter file codes; Catalog by file type, etc. Rewrite Error Messages—"Syntax Error" can be "Oops!" or anything you want!

All of Dos Boss's change features may be appended to your programs, so that anyone using your disks on any Apple (booted or not) will be formatting DOS the way you designed it!

PLUS . . . APPLE TIP BOOK NUMBER TWO!
With valuable DOS information: "Poking Around in DOS," "Making Programs Unlistable," "Two-Sided Disk Tips," "Care of Your Drives," "Creative Peeking & Poking," "3.2 vs 3.3," etc.

ONLY $24. +$1.50 Shipping
✓ **Dos Boss** on Applesoft Disk
✓ 40 page Beagle Bros Apple Tip Book #2
✓ Apple PEEKS, POKES & POINTERS Chart
MINIMUM 32K REQUIRED

AT YOUR APPLE DEALER NOW OR ORDER BY PHONE AT ANY HOUR... TOLL FREE!

Ask around about Beagle Bros Software.
Our programs are practical, friendly & user-oriented, and always come with an amazing amount of documentation & juicy Apple information. If your dealer doesn't have our stuff, get on his case about it, or order directly from us. We ship all disks & books WITHIN 48 HOURS.

BONUS WITH EACH DISK!
POKE your Apple all night long with this handy 11 x 17 reference poster! The most useable PEEKS, POKES, POINTERS & CALLS scrounged up from every source imaginable! A great companion to our original Apple Command Chart.

24 hour TOLL FREE Order Desk:
MasterCard, VISA or COD orders, call:
Nationwide: **800-854-2003** ext. 827
California: **800-522-1500** ext. 827
Alaska/Hawaii: **800-854-2622** ext. 827
(Orders only please. Questions, phone 714-296-6400.) COD orders, add $3.)

Or order by mail (use this coupon or separate sheet)

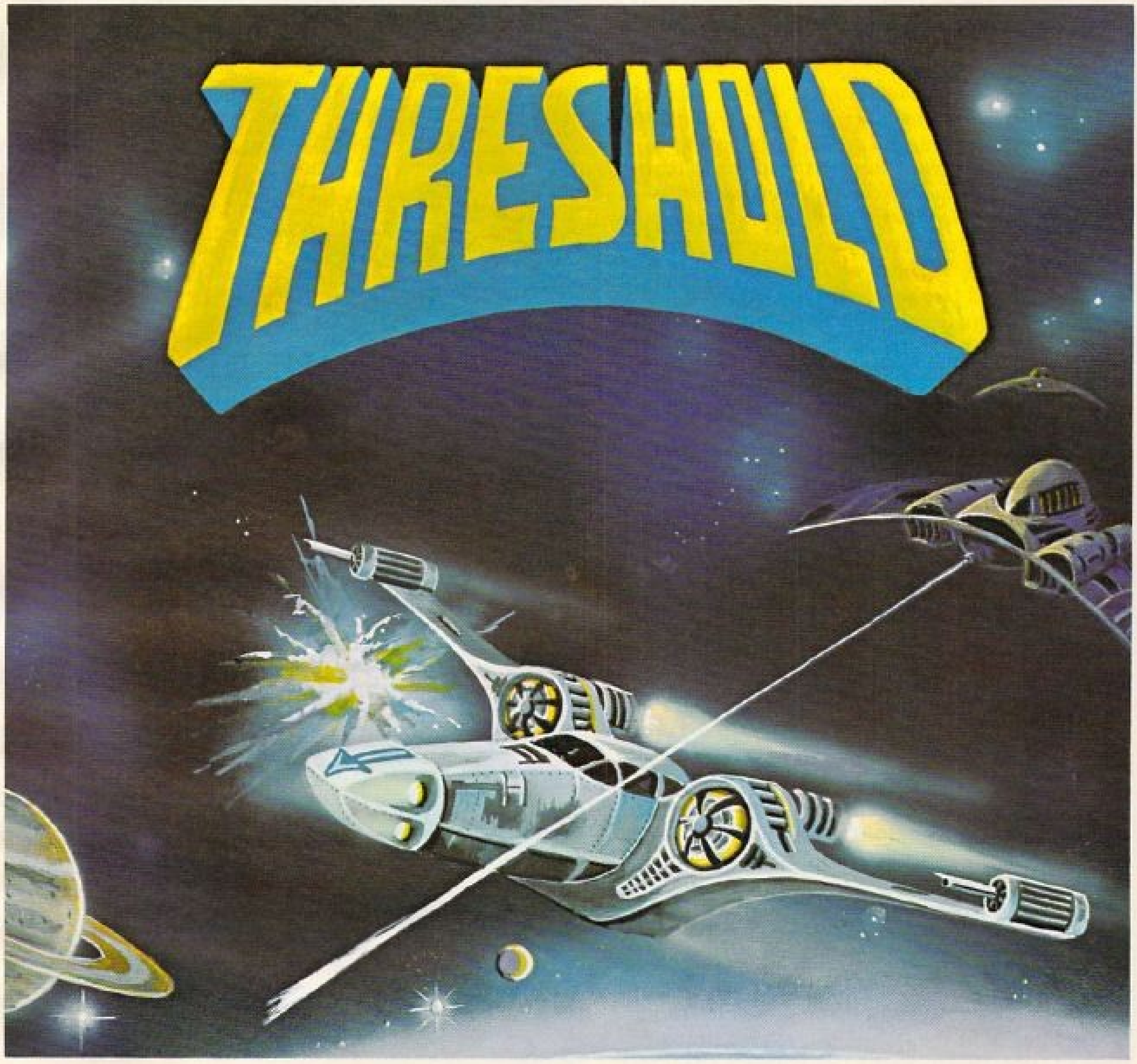
RUSH! The disks checked below
Plus The Tip Book and PEEKS & POKES Chart!
☐ **UTILITY CITY** ($31) ☐ **ALPHA PLOT** ($41) ☐ **DOS BOSS** ($25.50)
(Prices include First Class Shipping. Calif. residents add 6%.)

Specify
NAME$= ________________ ☐ **3.2** ☐ **3.3**
ADDR$= ________________
CITY$= ________________ **ZIP=** ______

VISA/MASTERCARD: Include Account No. & Expiration Date.

Mail U.S. check, money order or Visa/MC Numbers to: **BEAGLE BROS, Dept. 5**
4315 Sierra Vista,
San Diego, CA 92103
Telephone (714) 296-6400.

Please add $4.50 for shipping outside US or Canada. COD orders, add $3. California residents, add 6%.



ON-LINE SYSTEMS introduces arcade gaming as an art form. THRESHOLD, by WARREN SCHWADER and KEN WILLIAMS, features fast smooth animation, HI-RES graphics, and more challenge than you'll find in any other arcade game on the market.

THRESHOLD is an arcade game with alien attackers galore. In fact, there are more monsters out there than we expect you'll ever see. THRESHOLD was designed to be an arcade game that you won't get bored of, and that means a constantly changing game with a graduated skill level, but more than that, THRESHOLD means constant fun.

THRESHOLD runs on any 48K Apple II or II Plus DOS 3.2 or 3.3. Available now for $39.95 on disk from your local computer store or you may order directly from . . . . .

Now Available For ATARI 400/800 with 40K and a Disk Drive

ON-LINE systems

36575 Mudge Ranch Road - Coarsegold, CA 93614 - 209-683-6858

ADD $1.00 FOR DIRECT ORDERS

Visa, Master Charge, C.O.D. or Checks Accepted

The Beginner's Page

The ASCII Code

Richard Mansfield
Assistant Editor

It's easy to see how a typewriter puts a letter of the alphabet onto a piece of paper. You press the "F" key and the paper is struck by an inked ribbon, pushed against the paper by a small metal image of F. But what happens when you hit the F on a computer? It puts the number 70 into one of its memory cells.

What does 70 have to do with *F*? To answer that, we'll need to know what the ASCII Code is and learn the meanings of two BASIC commands: CHR$ and ASC.

Type in this short program:

```
 5 DIM A$(1) : REM THIS LINE IS ONLY
   NECESSARY ON ATARI
10 INPUT A$
20 PRINT ASC(A$)
30 GOTO 10
```

When you RUN this, you can type letters on your keyboard and see them translated into numbers. (Hit the RETURN key after each one.) Try "F" and you'll get 70. What you are seeing is called the ASCII Code. Computers only store *numbers* in their memory cells. In fact, they can only store the number one and the number zero. (For a more detailed explanation of how computers remember things, see "The Beginner's Page," **COMPUTE!** March, 1982, #22.) The computer can store words and symbols or pictures or anything else in patterns of these ones and zeros.

To store the letters of the alphabet, symbols like the percent sign, punctuation marks – all the keys on your keyboard – the computer uses a special code, the American Standard Code for Information Interchange, ASCII.

When Seven Is Not Seven

If you are RUNning the program above, type the number seven on your keyboard. It's not seven! In the ASCII code, it's 55. The number six, though, is 54, so the scheme is not entirely random. Why didn't they just use the number seven to stand for number seven in this code?

There are reasons for everything. If you learn to program in *machine language,* as opposed to BASIC, you'll work with hexadecimal numbers. In hex, the ASCII code for zero is 30 and seven is 37. In hex, it makes a bit more sense.

But back to ASCII. ASC, of course, is short for ASCII and you can find out what the ASCII equivalent of a single character is by typing PRINT ASC("F") or by asking for the ASC of a string variable (as we do with A$ in the program above).

You can go the other way with CHR$. This is BASIC's "character string" command. Where ASC translates a character into ASCII, CHR$ translates ASCII back into a character. So, you give CHR$ a number between 0 and 255 and it will give you a character. Here's a short program to see how CHR$ works:

```
10 INPUT X
20 PRINT CHR$(X)
30 GOTO 10
```

Each manufacturer has deviated somewhat from standard ASCII. For example, the Atari uses a code called ATASCII which is very similar to ASCII, but there are some differences. The creators of the ASCII standard had decided that the number seven should not print any character. Instead, seven is supposed to ring a bell, or a buzzer, or whatever sound your computer can make that can be used like a bell on standard typewriters. But on the Atari, if you type PRINT CHR$(7), the computer puts a graphics symbol, a large backslash, on screen. To sound the buzzer, use PRINT CHR$(253). By the way, CHR$(7) *does* ring a bell on Commodore computers with built in sound.

These variations on ASCII between computer models are one of the reasons that you cannot take a game on tape for the Apple and LOAD it into your PET.

The total of all 255 possible characters, graphics symbols, and buzzers that your computer can use is called its *character set.* To see your computer's character set, type in the following program:

```
10 FOR I=0 TO 255
20 PRINT CHR$(I);
30 NEXT I
```

The semicolon makes sure that they are put one after another on the screen (PRINT causes a carriage return unless the semicolon is there). But wait, what about the carriage return itself? Isn't it one of the ASCII code numbers? You bet. So are cursor-moving keys, reverse field, and other special tricks your computer can perform. When you RUN this program, get set for some strange effects. As the program runs through the numbers from zero to 255, it will encounter the clear-the-screen character too.

There are a number of uses for ASC and CHR$. With CHR$, you can send characters to your computer that cannot be typed in from the

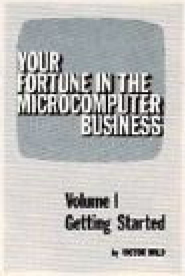
STARTING YOUR OWN MICROCOMPUTER BUSINESS

Starting your own microcomputer business is easy if you know the right steps to take. Two volumes of the new book **Your Fortune In The Microcomputer Business** describe the things you should know to start right and to build your business successfully.

Many people have good ideas for a successful microcomputer business. But they don't know how to put their ideas into action. These people are plenty smart. That's not the problem. The problem is in knowing simple things: How to analyze the market. How to select the right product or service. How to get enough startup money. How to plan your growth and success. The answers to these problems are not hard to understand. All you need is the right information. **Your Fortune In The Microcomputer Business** gives you the knowledge tools to start right, grow and prosper.

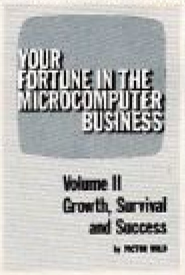
This practical reference manual has no hocus-pocus. It **does not** tell you how to do astrology or dating services. It **does** give you clear, complete, step-by-step instructions on how to get started right and insure your success!

Volume I Includes:



- The hottest trends in the market
- The seven best test markets
- How to read between the lines at a trade show
- 107 ways to reach your market
- Examples of the best ads in the business
- How to get free advertising
- The 21 steps to set up your business
- How to start a manufacturing business
- How to write a newsletter
- How to give a seminar
- How to start a service business
- How to package software for the mass market
- How to be a highly paid consultant
- Franchising—the good, the bad and the ugly
- Interviews with six successful microcomputer businesses
- How to use the RLC factor to be street smart, lean, mean and successful

Volume II Includes:


YOUR FORTUNE IN THE MICROCOMPUTER BUSINESS
Volume II
Growth, Survival and Success

- Strategies for growth
- Straight facts on incorporation
- How to build your organization
- How to manage cash flow
- How to develop your accounting system
- How to manage employees, wages and salaries
- How to survive a cash crunch
- What to do if things get really bad
- How to maximize your profits
- How to grow by acquisition
- How to value your business
- How and when to sell your business

WRITE OR CALL—DON'T WAIT!

Wildfire Publishing, P.O. Box 420 – DZ
Carpinteria, CA 93013 Ph. (805) 684-1489

Please send the following book(s) by return mail. I understand if I want to return them **for any reason** within 30 days of receipt, I can do so and get a **prompt full refund**.

☐ Your Fortune In The Microcomputer Business Vol. I, Getting Started, $20.00. (Postpaid)

☐ Your Fortune In The Microcomputer Business Vol. II, Growth, Survival and Success, $20.00. (Postpaid)

Calif. residents please add $1.20 each sales tax.

Name ______________________

Street ______________________

City __________ State __________ Zip __________

Enclosed is __________ or charge my MasterCharge ☐ or Visa ☐

Card # ______________________

Ex. Dt. __________ MCInterbank # __________

Signature ______________________ © 1982 Wildfire Pub.

keyboard. There might be no key on your keyboard which rings the bell. Use CHR$. This can be done directly from the keyboard or as part of a BASIC program.

Another common problem is trying to print quotes. You can't just type PRINT "THE "BEST" COMPUTER" because the set of inside quotes around the word *best* will confuse the computer. Try it. It will think you are printing the words *the* and *computer* with a numeric variable (*best*) between them. It will print a zero since the variable *best* has no value. To achieve the result you want, type PRINT "THE " CHR$(34) "BEST" CHR$(34) "COMPUTER".

If there is a printer attached and "listening" (responding) to your computer, you can make it do a carriage return by typing PRINT CHR$(13). Or, if the printer has a bell, try PRINT CHR$(7). Most printers accept the standard ASCII code and their instruction booklets will usually explain what numbers to send to perform backspace, underlining, etc. Remember, in this case it doesn't matter what codes your *computer* is using. When you send a letter to the printer, the *printer's* code (probably standard ASCII) will determine what gets put on paper. ©

LEARN

Having trouble learning to use your computer?

Reference manuals don't teach. Most BASIC texts don't cover specific personal computers.

TIS solves these problems with step-by-step books tailored for your machine.

For PET/CBM

Understanding Your PET/CBM Vol 1: Basic Programming	$16.95
PET Graphics	$ 6.95

For OSI C1P/C4P

Understanding Your C1P/C4P A Workbook of BASIC Exercises	$ 9.95

For VIC

Understanding Your VIC Vol. 1: Basic Programming	$13.95

Money Back Guarantee. VISA/MC accepted.
All prices include UPS or 1st Class postage.

TIS INC

Total Information Services, Inc.
Box 921, Dept. C
Los Alamos, NM 87544

Sams helps you cut deeper into the workings of your Apple with two new additions to our crop of Apple® Computer Books.

APPLE® FORTRAN by George Blackwood is a beginner's guide to using FORTRAN, a powerful, science-oriented language, on your APPLE® computer. NO. 21911, $12.95

CIRCUIT DESIGN PROGRAMS FOR THE APPLE II® by Howard M. Berlin lists several BASIC-language programs which simplify the design and analysis of commonly encountered electronic circuits. NO. 21863, $15.95

The Sams Apple® collection also includes: APPLE® INTERFACING with tested designs for interfacing circuits and their associated BASIC software, allowing you to connect your APPLE® computer to the real world. No. 21862, $10.95

APPLESOFT LANGUAGE, NO. 21811, $10.95 is a beginner's guide to APPLESOFT, the native language of the popular APPLE II Plus® microcomputer. INTIMATE INSTRUCTIONS IN INTEGER BASIC introduces you to the fundamental language of the standard APPLE II®. NO. 21812, $7.95

Sams continues to slice through the Apple®, giving you a good look at some of the very best pieces of this powerful machine.

To order these Sams Books or to get the name of your Sams retailer, call 800-428-3696 toll free, or 317-298-5566 and reference AD180.

SAMS BOOKS ™
Howard W. Sams & Co., Inc.
4300 West 62nd Street, P.O. Box 7092
Indianapolis, IN 46206

NEW SLICES FOR YOUR APPLE®

APPLE INTERFACING
APPLESOFT LANGUAGE
CIRCUIT DESIGN PROGRAMS FOR THE APPLE II
INTIMATE INSTRUCTIONS IN INTEGER BASIC

Offer good in USA only and expires 9/30/82. Prices subject to change without notice.
Visit Sams at the NCC and Comdex. Booth Nos. 8223 and 1710.

COMPUTE! has established review panels, each consisting of three or more reviewers whose backgrounds qualify them to analyze new software or hardware products.

To better prepare COMPUTE!'s readers for their buying decisions, and to present fair reviews, we collect the independent opinions of the panelists into one Overview. We think that you will find COMPUTE! Overviews complete, balanced, and informative.

Compilers take, for example, a BASIC program and translate it into a machine language-like form. It should then execute for faster than the original BASIC.

COMPUTE! Overview:

The Galfo Apple Compiler

The Integer BASIC Compiler from Galfo Systems contains two diskettes: one is a system diskette and the other is a compiler diskette.

Compilers are being marketed today primarily because BASIC is slow in execution speed – painfully slow for some applications. This is so because each BASIC command or instruction in a program must be converted by the interpreter before it can be understood and processed by the CPU (central processing unit). This conversion must be done each time the command or instruction is encountered in the program, thus contributing to the slow speed of a BASIC program. Compilers enjoy their speed advantage because these conversions are done only once – *before* the finished (or *compiled*) program is run. After the BASIC program is compiled, it will execute like a machine language program. The Galfo compiler creates code that is comparable to machine language in execution time. It will deliver truly fast programs.

There are several important and desirable features to look for when selecting a compiler. Can the compiled program be stored in any portion of memory desired? With what type of program information is the user furnished after the program is compiled? What is the speed advantage of the compiler? Is the compiled program (object code) longer (and if so, how much longer) than the original program (source code)? How good is the error-handling capability of the compiler? Is it easy to use? Is the documentation complete and easy to follow? Will the compiler handle the many different types of routines demanded by the user, such as graphics, string handling, and I/O routines?

The Galfo compiler allows users of Integer BASIC to make their programs not only faster, but also smaller! This somewhat paradoxical situation is due to the fact that the Integer BASIC Compiler (or IBC) can produce two varieties of output at the user's option:

1. Pure GSL Code
2. Mixed GSL and 6502 Code

GSL stands for "Galfo Stack Language," and is the machine code for an idealized 6502 stack-oriented computer. (Compare Sweet-16 code, which is the machine code for an idealized 16-bit, register-oriented, 6502-like machine). The GSL generated by IBC for a typical Integer BASIC program is more compact than the corresponding tokenized internal form used by the Integer BASIC Interpreter. Even with the addition of 6502 code mixed with the GSL code, reasonably compact programs are achieved. Hence, smaller programs!

...allows users of Integer BASIC to make their programs not only faster, but also smaller!

A runtime system is used to execute the GSL object code. The GSL.SYS program is a Galfo Stack Computer Emulator (Again, compare this to the Sweet-16 program, which is a Sweet-16 Computer Emulator).

It does not have to perform any translation tasks such as the Integer BASIC Interpreter does – such as locating variables or line numbers by searching through memory, converting numbers from ASCII strings to binary, etc. Therefore, it executes the same BASIC program much faster than the Integer BASIC Interpreter can.

About the only overhead is the "fetch-execute" cycle in the emulation. This means retrieving the next Galfo Stack Computer opcode from the object code and dispatching a call to the appropriate subroutine to emulate that opcode. In programs which have a mixture of GSL code and 6502 machine code, this overhead is totally eliminated for those parts of the program in 6502 code. The upshot of all this is that the IBC compiled programs run from 7 to 50 times as fast as their interpreted counterparts. This is the claim made by the documentation. Let's examine the claim.

Benchmarks

The Instruction Manual provided with the compiler contains an Appendix listing some benchmark data. The programs tested to provide that data were obtained from articles appearing in *Kilobaud* magazine, in 1977. The measured speeds for those programs using the normal Integer BASIC Interpreter ranged from 1.4 to 28 seconds. The Appendix lists two corresponding sets of numbers for the same programs when compiled under IBC. One set measures them as compiled to pure GSL code, and another set measures them as compiled to mixed GSL and 6502 code. The data and one of the benchmarks are shown in Figure 1.

Figure 1.

Program #	IBC/GSL (Opt. speed)	IBC/GSL (Opt. space)	APPLE Integer BASIC	APPLE Applesoft BASIC
BM1	0.16	0.16	1.4	1.3
BM2	0.33	0.46	3.2	8.0
BM3	1.5	1.8	8.0	16
BM4	1.0	1.2	7.0	17
BM5	1.2	1.3	9.0	19
BM6	2.1	2.3	18	28
BM7	2.9	3.4	28	45

```
Listing of BM7:

   300 PRINT "START"
   400 K=0
   430 DIM M(5)
   500 K=K+1
   510 A=K/2*3+4-5
   520 GOSUB 820
   530 FOR L=1 TO 5
   535 M(L)=A
   540 NEXT L
   600 IF K<1000 THEN 500
   700 PRINT "END"
   800 END
   820 RETURN
```

BM7 was compiled in an attempt to verify the claimed data. The data were gathered using a stop watch, and may not be as accurate as data obtained on a system with a realtime clock. (The Instruction Manual does not comment on how its data were obtained).

These results show that IBC is able to produce most efficient code which really zips along. In fact, the author's suggestion that IBC is the fastest 6502-based high-level language just may be accurate.

Other Speed Tests

A compiler gains much of its speed because there is no longer any need for interpreting each statement. This means different programs will speed up by differing amounts. For example, compiling a FOR-NEXT loop containing the multiplication of decimal numbers will not speed up very much, as most of the time is used in the multiplication. The same loop multiplying Integers will undergo a great increase in speed.

The compiler was also tested using the Benchmark program from *Call-Apple*, March/April, 1980, with the loops increased to 10000. The Benchmark programs do the following: 1. Simple FOR/NEXT loop 2. IF/THEN loop 3. compute using loop variable 4. compute using constants 5.GOSUB 6. GOSUB with additional loop 7. Storing a variable in an array.

The compiler can compile either for speed or space. The speed difference between the two was about 10%. The increase in speed compared to normal Integer BASIC is tremendous. Time is in seconds. Again, timing was done with a stop watch and not a built-in clock. The shorter times are, therefore, somewhat inaccurate:

TEST #	1	2	3	4	5	6	7
INTEGER BASIC	14	35	81	46	130	245	360
COMPILED FOR SPACE	2	6	19	6	14	24	34
COMPILED FOR SPEED	2	4	17	4	12	22	30

Compile time for the Benchmark program was about five seconds.

The compiler was tested under actual conditions. Four pre-existing Integer BASIC programs were compiled using both GSL code (conserving disk space) and mixed code (for optimum program speed). One of the programs compiled was a pinball game. Have you ever tried to play computer-pinball while the ball literally flies across the screen? The compiled program ran so fast that it was impossible to play the game. A second Integer program that played a musical tune was compiled. The individual notes sounded in such rapid succession that the tune played like a continuous musical slur. The individual notes no longer seemed to be separate notes. The time required to *compile* these programs was of such short duration that it bears little mention. Suffice to say that a typical 300 line program compiled in seconds!

There were a few surprises in store when comparing the disk space used by a compiled program to the original program's disk space. When compiling for maximum speed, disk space used increased from a low of 29% to a high of 58% more than the original program. When compiling for maximum economy in disk space, the space used by the compiled program actually *decreased* in every case! Space *saved* ranged from a low of 20%

to a high of 40% of the original program's disk space usage. Additionally, programs compiled for disk space ran almost as fast as those compiled for speed.

The compiler can be used with one disk drive, but this creates an awkward operating situation. It is recommended that the compiler be utilized in a two-drive system.

Using The Compiler

Space does not permit a complete step-by-step description of the compilation process, so the most important procedures and features will be described. Always begin operating by booting the *system* diskette in drive 1. This automatically loads the routines necessary for proper operation of the compiler. It is suggested that a cold start be effected if any utility programs are present in RAM such as the Program Line Editor. Various problems were encountered while attempting to compile with utility programs in RAM. Compiling after a cold start eliminates these problems.

After the system diskette is booted, load the program you wish to compile in the usual manner (from drive 2). Replace your program diskette with the *compiler* diskette and type "BRUN IBC,D1". This command begins the compilation process. The first prompt you will see asks whether you wish to compile using execution speed or disk-space as a priority. This compiler produces code that is *so* fast and so compact, that this option usually becomes a relatively minor consideration. The resultant object code runs almost as fast, and in many cases *as fast*, as a compilation for speed.

The user is then given a number of additional options such as choosing a starting address for the object code and executing or saving the program on disk (as a binary file). An excellent feature is that the object code produced by this compiler will run on *any* Apple II computer. This means that an Integer BASIC program compiled with the Galfo compiler will run on an Apple II + (a system without Integer BASIC in ROM). This feature was tested, and the code does indeed execute problem-free.

There is a short (3K) program entitled, "GSL.SYS" which must be present on your diskette in order to run the object code (compiled program). The procedure is simple. Using Apple's FID copy program, copy the GSL.SYS program from the "system" diskette to your program diskette. Any compiled program that is BRUN from your program diskette will automatically look for, and then load and run, the GSL. The GSL therefore, must be resident on the program diskette. The GSL program loads at $8800 in memory by default. An advantage noted, when comparing this compiler to others, is that the comparable programs used with the other compilers become an integral part of *each* compiled program. This unnecessarily increases program size and, consequentially, decreases available space on the diskette.

Added Features

Integer BASIC restricts the length of strings to 255 characters. A string length of 32767 is permissible with the Galfo compiler. A symbol table is presented to the user after compilation, listing all variables encountered in the program, the type of variable, (string, integer, or array) and the location of each in memory. A method is provided whereby the user can trick the computer and cause two variables to share the same memory location. This enables the user to refer to either variable during the course of a program and yet retrieve the same information from the variable table. The method used is simple, fully described in the manual, and can easily be managed by even a novice programmer.

The compiler provides the user with many new and modified commands, too numerous to be fully documented in this review. These commands are implemented by typing DSP before each command. Some of the commands that can follow the DSP prefix are: HOME, CLEAR, INVERT, FLASH, NRML, FULL, MIXED, LO, HI, H2, POINT, LINE, and SHAPE. These commands control printing to the CRT, and graphics implementation. If you inspect the commands, you will realize that they emulate the commands available to you in Applesoft. These commands are especially useful as they eliminate the need for the usual cumbersome POKEs (their counterparts in Integer BASIC). The author of the program has ingeniously reconfigured Apple's DSP command, and used it to his advantage for the special operatives.

When writing an original program for compilation or when converting an existing one, the user should be aware of two potential trouble spots. DIM statements *must* be defined by using integers, as the compiler will not function with variables in DIM statements. Additionally, variables used in GOTO's or GOSUB's will cause a large increase in program size. This is because the compiler builds a variable table which is searched each time a variable is encountered in a GOSUB or GOTO statement.

Compiled programs are callable from either Integer or Applesoft. However, after running the program by calling it from Applesoft BASIC, strange errors occur unless the memory pointers are reset by doing an FP. Whether calling compiled Integer BASIC programs from Applesoft is a useful capability is difficult to say. The license to use the run-time system in programs for sale is stated as $5. If this is a one-time fee, it is reasonable. It is not clear whether this fee is for each program sold or is a sub-license fee.

A Few Minor Caveats

The compiler enables the TRACE mode while compiling, and the computer is left in this mode after the compilation is done. The user must then manually execute the NOTRACE command. This may seem like a trivial problem until you run the program only to have line numbers begin printing on the CRT each time. Also, there is no provision made whereby the user may protect a given portion of memory. For example, if one wished to protect the HI-RES page from being overwritten, the only way found to easily effect this was to load the entire object code *after* the HI-RES page in memory.

Several errors of omission were encountered, although none were serious. For example, a file entitled "HI-RES Driver" is needed to utilize the HI-RES routines supplied. The manual states that the file must be *loaded* for the routines to work, while in actuality it must be *run*. Additionally, no mention is made as to how one may automatically effect this. If Apple's HELLO program is used to BRUN the HI-RES Driver, then, after the HI-RES file is loaded, the program will stop. An EXEC file must be created in order to first BRUN the HI-RES Driver and then BRUN your own compiled program. No mention is made of this in the manual.

General Overview

• Panelist #1: "Generally great. Look forward to Galfo Applesoft Compiler."

• Panelist #2: "For heavy users of Integer BASIC, the IBC is well worth having. It should enhance existing programs as well as open many new avenues of application that were formerly unavailable for reasons of performance."

• Panelist #3: "The Galfo Integer BASIC Compiler is an extremely useful utility. Its advantages and features far outweigh its shortcomings (which are few) and future revisions will almost certainly correct these. Source code is compiled extremely fast and object code produced by this compiler executes almost as fast as pure machine language. This is a package that all Integer BASIC programmers should own."

Galfo Integer BASIC Compiler. Galfo Systems, 6252 Camino Verde, San Jose, CA, 95119. $149.50. ©

Atari is a reg. ™ of Atari, Inc.

UPGRADE YOUR ATARI®

REPORT CARD

STUDENT Basic A+
PARENT Optimized Systems Software

SUBJECTS

ADVANCED I/O	A+
STRUCTURED PROGRAMMING	A+
PLAYER/MISSILE GRAPHICS	A+

DEPORTMENT

EASE OF USE	A+
FLEXIBILITY	A+
COMPATIBILITY	A+

THE ONLY LOGICAL UPGRADE FOR ATARI BASIC!

GRADUATE TO BASIC A+ FROM THE AUTHORS OF ATARI BASIC!

BASIC A+

BASIC A+ will rate an A+ from any Atari user! Upward compatible with Atari Basic, it adds statements and features that enhance the Atari 800's real power, flexibility, and ease of use: Superior I/O features for business and other applications. Additional file manipulation commands. Significant help in program development and debug. Structured programming aids. And MORE! A partial list of the enhancements of BASIC A+ includes:

RPUT/RGET (record I/O) **BPUT/BGET** (binary I/O) **ERASE PRINT USING SET TAB INPUT"..." DIR PROTECT RENAME TRACE WHILE...ENDWHILE IF...ELSE...ENDIF SUPERB PLAYER/MISSILE GRAPHICS**

BASIC A+ requires a disk and 32K bytes of RAM. Since no cartridge is used, BASIC A+ will take advantage of all the RAM (48K bytes) in a maximum Atari 800 system (recommended) $80

OS/A+

Completely compatible with Atari's DOS (version 2), but with an advanced, command-driven console processor. Simple. Flexible. Powerful. With an easy-to-use **BATCH** capability. OS/A+ **INCLUDES** all the following utilities (and more):

EASMD (Editor/ASseMbler/Debug) is our upgraded all-in-one assembly language development package for the 6502 microprocessor. The editor — with such features as FIND and REPLACE — can even edit BASIC A+ programs. The assembler can include multiple source files in a single assembly.

DUPDSK and FORMAT create master or slave disks. Make sector by sector copies of any OS/A+ disk.

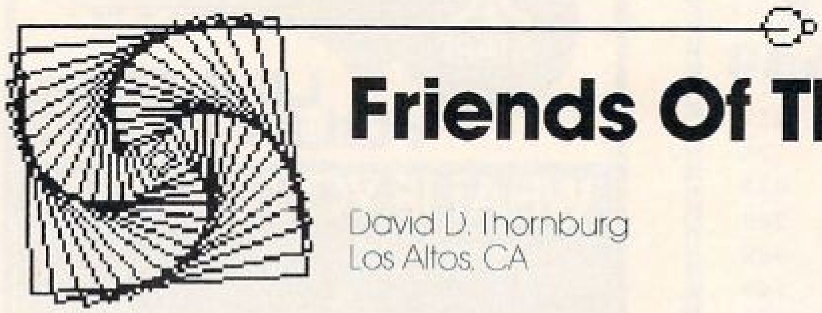
COPY a simple, single file copy utility.

All of this power is included in our OS/A+ package $80

For the utmost in capability and flexibility, our combination system — **BASIC A+ and OS/A+** — is available $150

SOFTWARE IN A CLASS BY ITSELF
AVAILABLE NOW THROUGH LOCAL DEALERS

Optimized Systems Software, Inc.
10379 Lansdale Ave., Cupertino, CA 95014,
(408) 446-3099



Friends Of The Turtle

David D. Thornburg
Los Altos, CA

Your letters keep pouring in, and I am writing answers as fast as possible. Many of you have asked about the availability of Apple LOGO. There are rumored to be several versions of this language which either are, or will be, available from several vendors (it is risky writing columns a month or so in advance in a field as active as this one). The version available from Apple is a product of Logo Computer Systems Inc. I have looked at it and like it very much. I also saw two draft manuals – one reference manual and one introductory manual which introduces LOGO through the use of the turtle. Readers with 64K Apples will be most impressed with this language.

COMPUTE! reader Thomas Granvold wrote to tell me that he and his wife use Atari WSFN's turtle graphics to help them in quilt design. His wife selected the pattern resulting from the following procedure for use in a quilt pattern:

```
HCN8 (R8(R15F))
```

This procedure draws eight octagons around a common point. These will then be put in a square which circumscribes the octagon to form the basis for the quilt pattern. For those of you who don't understand the alphabet soup of WSFN, the above procedure translates as follows: "Home, Clear, point North, repeat 8 times the task of turning right by 45 degrees and 8 times turning right by 45 degrees and drawing a line 15 units long." In WSFN the R command turns the turtle to the right by 45 degrees and the F command moves the turtle forward by one unit.

Randolph Schleef of the Miami-Dade Community College (11380 NW 27 Ave., Miami, FL 33167) is particularly interested in the use of turtle geometry with disabled users. Any of you who have information concerning applications in this area should contact him directly.

Is This Trip Really Necessary?

This winter I had the opportunity to teach computer programming to children at a local school. Each week I spent one-half hour in each class from second through sixth grade. The principal teaching tool was (of course) the turtle – both in the form of Atari PILOT and the Milton Bradley Big Trak. During these classes I was delighted to see that children have a pretty accurate intuitive sense when it comes to geometry. One of my favorite geometrical rules is called the Total Trip Theorem. This theorem states that if you send the turtle off to trace out a simple closed path, that by the time the turtle has returned to its original location and orientation it will have turned by exactly 360 degrees.

A few examples should convince you that this is a plausible idea. Suppose we first have the turtle trace out a square. In Apple LOGO, we would type:

```
REPEAT 4 [FORWARD 30 RIGHT 90]
```

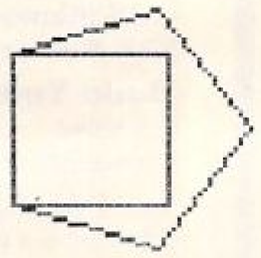
to draw a square 30 units on a side.

```
REPEAT 4 (FORWARD 30 RIGHT 90)
```

Figure 1.

Notice that in drawing this square, the turtle turned 90 degrees four times, or 360 degrees overall. To draw a pentagonal path with the same length sides, we would type:

```
REPEAT 5 [FORWARD 30 RIGHT 72]
```



```
REPEAT 4 (FORWARD 30 RIGHT 90)
REPEAT 5 (FORWARD 30 RIGHT 72)
```

Figure 2.

COLLEGE BOARD SAT* PREP SERIES

TRS-80, APPLE, PET, OSI, ATARI, CP/M, PDP-11

Each program confronts the user with a virtually limitless series of questions and answers. Each is based on past exams and presents material on the same level of difficulty and in the same form used in the S.A.T. Scoring is provided in accordance with the formula used by College Boards.

S.A.T., P.S.A.T., N.M.S.Q.T. — Educator Edition set includes 25 programs covering Vocabulary, Word Relationships, Reading Comprehension, Sentence Completion, and Mathematics. **Independent tests of S.A.T. series performance show a mean total increase of 70 points in students' scores.** Price **$229.95**

GRADUATE RECORD EXAM Series - Educator Edition includes 29 programs covering Vocabulary, Word Relationships, Reading Comprehension, Sentence Completion, Mathematics, Analytical Reasoning and Logical Diagrams. Price **$289.95**

COMPETENCY PROFICIENCY EXAM PREP SERIES

This comprehensive set of programs consists of simulated exam modules, a thorough diagnostic package, and a complete set of instructional programs. It is designed to teach concepts and operations, provide drill and practice and assess achievement levels through pre and post testing. The Competency Exam Preparation Series provides a structured, sequential, curriculum encompassing mathematical, reading and writing instruction.

This program is designed for individual student use or use in a classroom setting. Programs provide optional printer capability covering worksheet generation and performance monitoring. C.E.P.S. are available in three software formats.

Special editions available for California Proficiency Assesment Test and New York Regents Competency Tests. Call for Prices.

Krell's M.I.T. Logo for Apple* $179.95

Complete 4 disk set includes Library Disk, 20 program Tutorial Disk, **Alice In Logoland** and **2 COPIES OF M.I.T. LOGO FOR APPLE***

Odyssey In Time

This spectacular adventure game adds a new dimension of excitement and complexity to **Time Traveler.**

Odyssey In Time includes all the challenges of **Time Traveler** plus 10 additional eras. Each game is different and may be interrupted and saved at any point for later play. **$39.95**

Time Traveler

The best of the adventure games. Confronts the player with complex decision situations and the demand for real time action. Using the Time Machine, players face a challenging series of historical environments. To succeed you must build alliances and struggle with the ruling power. Each game is unique. **$24.95**

Isaac Newton +

Perhaps the most fascinating and valuable educational game ever devised - **ISAAC NEWTON** challenges the players (1-4) to assemble evidence and discern the underlying "Laws of Nature" that have produced this evidence. **ISAAC NEWTON** is an inductive game that allows players to intervene actively by proposing experiments to determine if new data conform to the "Laws of Nature" in question. Players may set the level of difficulty from simple to fiendishly complex.

F.G. Newton

Full Graphics Newton. This version of Isaac Newton presents all data in graphic form. Because data is graphic rather than symbolic, this game is suitable for very young children. Players may select difficulty levels challenging to the most skilled adults. **$49.95**

☆NEW☆ ☆MICRO-DEUTSCH☆

Micro-Deutsch set includes 24 grammar lessons, covering all material of an introductory German course. Four test units also included. Grammar lessons use substitution transformation drills, item ordering, translations and verb drills. Drill vocabulary based on frequency lists. Suitable for use with any high school or college textbook. Extensively field tested at SUNY Stony Brook. Available for Apple II and PET/CBM. (PET version includes a special foreign language character chip.) Also available soon: MICRO-FRANCAIS, MICRO-ESPANOL, MICRO-IVRIT, MICRO-YIDDISH, MICRO-CHINESE, MICRO-JAPANESE. **$179.95**

☆ NEW ☆ ☆Pythagoras and The Dragon☆

Mathematics in a fantasy game context. Based on **The Sword of Zedek, Pythagoras and The Dragon** introduces Pythagoras as a mentor to the player. When called on for aid, Pythagoras poses math questions, and depending on the speed and accuracy of the player response, confers secret information. With Pythagoras as an ally, the quest to overthrow Ra, The Master of Evil, assumes a new dimension of complexity. Depending on the level chosen, problems range from arithmetic through plane geometry.

32K **$39.95**

Free Bonus with purchase of $300.00 or more: Applesoft Tutor Series

Krell Software Corp. has no official ties with the College Entrance Examination Board or the Educational Testing Service. Krell is, however, a supplier of products to the E.T.S.

PROGRAMS AVAILABLE FOR TRS-80, APPLE II, PET & ATARI

N.Y.S. residents add sales tax.

All programs require 16K ● TRS-80 programs require LEVEL II BASIC ● APPLE programs require Applesoft BASIC

*Trademark Apple Computer Corp.

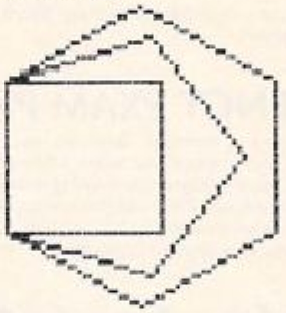
KRELL SOFTWARE CORP.

"The State of the Art in Educational Computing"

21 Millbrook Drive, Stony Brook, N.Y. 11790 (516) 751-5139

For this figure, the turtle turned 72 degrees five times, or 360 degrees overall. Finally, let's look at a hexagonal path:

```
REPEAT 6 [FORWARD 30 RIGHT 60]
```



```
REPEAT 4 [FORWARD 30 RIGHT 90]
REPEAT 5 [FORWARD 30 RIGHT 72]
REPEAT 6 [FORWARD 30 RIGHT 60]
```

Figure 3.

It should come as no surprise to find that the total amount turned is once again 360 degrees.

Abelson and diSessa cover this topic quite thoroughly in their book *Turtle Geometry;* and, as I said before, grade school children seem to have an intuitive feel for this result.

But if this theorem works well for a turtle walking on a flat surface, how does it work for a turtle walking on another surface – say that of a cube?

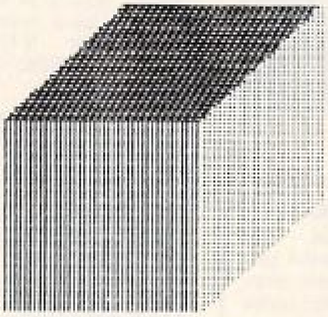


Figure 4.

Suppose the turtle starts off at the center of the front face and is pointing up. We could have the turtle walk forward to the center of the top face, being careful to walk straight over the edge without turning.

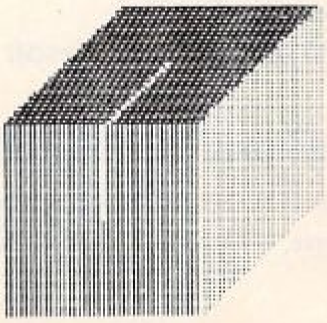


Figure 5.

Next, we can have the turtle turn to the right by 90 degrees and walk in a straight line to the center of the right face.

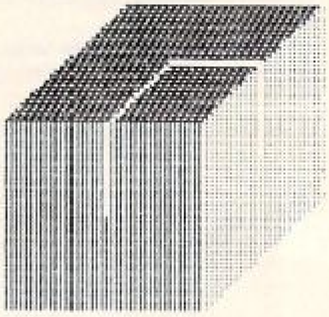


Figure 6.

Once the turtle has arrived there, the turtle once again turns to the right by 90 degrees and walks in a straight line to the center of the front face. When it turns right by 90 degrees, it is then back at its starting location and direction.

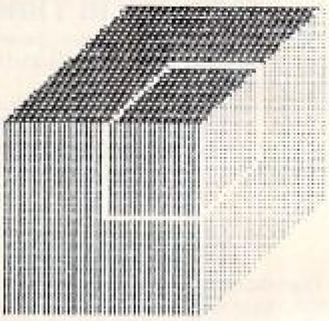


Figure 7.

If you have been keeping up with the number of turns, you have probably noticed that our turtle has made a nice closed path but has only turned 270 degrees. What happened to the missing 90 degrees?

To see what happened, we only need to spread part of the cube out into a flat surface (since we know what turtles do on flat surfaces). If we fold the front and right face up to be on the same surface as the top of the cube we see this picture:

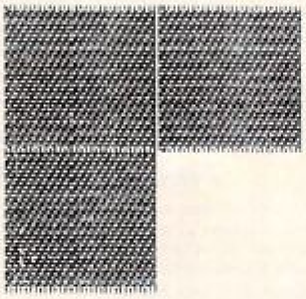


Figure 8.

Now let's have the turtle trace its path once again. Seeing what happens with the first few steps is pretty simple. The turtle moves forward and turns right twice.

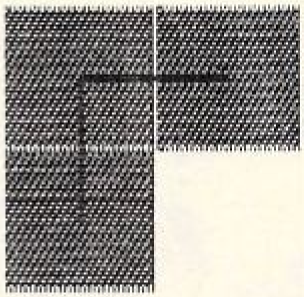


Figure 9.

Next, the turtle has to get back to its starting position. Since the cube was opened flat, the edges of the front and right side which normally touch are now spread apart. As you can see, they are spread

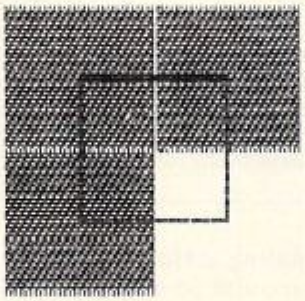


Figure 10.

apart by 90 degrees. So if we want to complete the closed path, we have to turn the turtle the extra 90 degrees to make it connect with its starting point. This shows that, if the turtle isn't walking on a plane surface, the total turning angle for a closed path may not be 360 degrees. If the angle is some other value, then the difference between this value and 360 degrees is the size of the angular "gap" that would be created by spreading the curved surface flat.

I hope you find little excursions like this to be interesting. One of the nice features of turtle geometry is its ability to make some difficult mathematical concepts easy to see.

If you know any other of these types of illustrations (or would like me to find some more of my own) let me know what you have or want, and I will share the results with you all.

Once again, please keep me posted of your activities and interests with turtles so that I can share them with your fellow Friends of the Turtle.

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

©

EDUCATIONAL SOFTWARE FOR CHILDREN

- Wide selection of 8K and 16K programs
- Programmed with motivational graphics and sound
- Money back guarantee

ORDER YOUR FREE CATALOG AND . . .

Expect more from your PET®.

PET is a registered trademark of Commodore Business Machines.

MICROGRAMS INCORPORATED
P.O. BOX 2146, LOVES PARK, IL 61130
PHONE 815/965-2464

☐ Please send me a free K-8 catalog.

☐ Please send me a sample program and a free K-8 catalog. I have enclosed $2.00 for postage and handling.

NAME ______
ADDRESS ______
CITY ______
STATE ______ ZIP ______

It is not immediately obvious, but if you know machine language and want to change some aspect of your ROM, you could move the code down into RAM and adjust the JSR's, JMP's, etc. to reflect the new locations. Then, you can modify the previously ROM-frozen program all you want. Here, the Apple II Plus Monitor is augmented by adding Step and Trace functions after moving it into RAM.

Softmon: Restoring Trace And Step To The Apple II Plus Monitor

R. Hiatt
Brock University
St. Catharines, Canada

A monitor trace is a useful tool for debugging machine code programs, in the same way that TRACE works for BASIC. In addition, it is invaluable for deciphering commercial binary routines which tend to be convoluted and to rewrite parts of themselves so that a disassembled listing taken at one point of a run will differ from that taken at another point.

The old Apple II *had* monitor trace, and the Apple II Handbook gives a disassembled listing of Steve Wozniak's monitor, resident at $F800-$FFFF, (as is the monitor of the current II Plus). Comparison of the new with old monitor shows relatively few differences, only 200 or so bytes of code. In particular, the subroutine STEP ($FA43-FAD6, $FAFD-FB1D) has been changed so that keyboard entry of 'AdrT' or 'AdrS' <CR> simply does an RTS.

The II-Plus monitor, being in ROM, can't be changed, of course, (and perhaps that's just as well). But it can be copied. In fact, it will obligingly copy itself. The procedure is as follows:

Key in:

1. CALL-151 <CR> (puts you into monitor)

2. 4800<F800.FFFFM <CR> (moves, *i.e.*, copies the code from the second address through the third address to a location starting at the first address.)

The move is so fast that it's hard to believe anything has happened, but you really do have a copy of the monitor in RAM, residing at $4800-4FFF, and *this* can be changed in any way you want. Moreover, it already works. Try it; leave monitor with RESET and execute a CALL 20329. The familiar star appears on the screen and you're in *soft monitor*.

The soft monitor appears to do everything that the resident monitor does. In fact, the resident monitor is still doing all the work. The only part of the soft monitor being used is the directory (now at $4F65-4FFF) which still calls subroutines in the $F800-FFFF area. Some changes have to be made. (If you're making changes on your own, it's expedient to make one change right away, and that's the prompt character. For example replace the $AA at $4F6A with $A3. This will produce a "#" instead of a "*". Without this, it's awfully easy to get back into resident monitor without knowing it.)

The changes I was interested in making are done by the BASIC program, Makesoftmon (Program 1). In fact, the program does the whole job, including the original monitor copy, since that subroutine can be called from BASIC after POKEing the appropriate addresses (lines 110-130). A simple loop (lines 160-190) detects the JSR's and JMP's that transfer control within monitor, and changes the high byte of the address from $Fx to $4x. A few discrete POKEs are necessary (lines 210-220) to change the high bytes of some indirect addresses and to restore the character table to old monitor form. Finally, the STEP subroutine is read from DATA statements and POKEd.

That completes Softmon. For the user who doesn't want the soft monitor at $4800-, but somewhere else, Makesoftmon is easy to change. It's just a matter of replacing all $4x high bytes with the high byte address desired.

Obtaining the STEP and TRACE functions via Softmon carries a fairly high overhead. Ideally, Softmon should call the resident monitor for those routines it handles and carry code only for those things it won't. But that's another project. At the moment, I'm more interested in unravelling some other mysteries using the newly acquired functions.

To use monitor TRACE and STEP with Softmon in memory ($4800-$4FFF), either CALL 20329 or CALL-151 and then execute 4F69G.

To execute a *trace,* key in the desired start address (in hex, of course), followed by "T," and then carriage return (CR). For STEP, key in start address followed by "S," CR. To go on to the next instruction, simply key "S," CR. Both trace and step display the disassembled instruction plus the contents of the A, X, Y, S and P registers, and most importantly, *perform* the instruction, so that branches are followed properly and new code is created.

A certain amount of discretion is involved with these. Trace scrolls the information up the CRT about twice as fast as a BASIC LIST. To see what it's doing in detail really requires a printer, and even then it's very easy to go through a ream of paper needlessly if trace encounters loops of any length. I'd be inclined to say the best use of trace would be in debugging your own programs on CRT only, to the extent of insuring that there are no infinite loops, and that the final RTS is correct.

Step has much more potential, the pace is user-controlled. If trapped in a long loop, one can simply step out by keying a step to the alternate branch. Better yet, rather than stepping out explicitly, monitor in the information that will cause the desired branch and then step the location again. This will ensure that subsequent step-traces will be getting correct code.

A cautionary note: since trace and step do perform the instructions, the same caution has to be observed as with any POKE or user-written machine code routine. Switches will be set or reset, *no* permanent damage can be done, but the system can be crashed or DOS can be disabled.

```
10  REM  MAKE SOFTMON
20 D$ =  CHR$ (13) +  CHR$ (4)
30  DEF  FN LB(D) = D - 256 *  INT (D / 256)
40  DEF  FN HB(D) =  INT (D / 256)
50  DEF  FN DC(I) =  PEEK (I) + 256 *  PEEK (I + 1)
60  REM  PAGE ADDRESSES IN $4800-4FFF BY HEX DIGIT
70 S8 = 18432:S9 = 18688:SA = 18944:SB = 19200:SC = 19456:SD = 19712:SE =
    19968:SF = 20224:SG = 20480
80  REM  A1,A2,A4 HIGH & LOW BYTES
90 L1 = 60:H1 = 61:L2 = 62:H2 = 63:L4 = 66:H4 = 67
100 MS = 63488:ME = 65535
110  PRINT "COPYING MONITOR INTO $4800-$4FFF"
120  POKE L1, FN LB(MS): POKE H1, FN HB(MS): POKE L2, FN LB(ME): POKE H2,
     FN HB(ME): POKE L4, FN LB(S8): POKE H4, FN HB(S8)
130  CALL 65068
140  PRINT "CHANGING ADDRESSES FOR JSR'S AND JMP'S"
150  PRINT "CHANGED ADDRESSES ARE LISTED"
160  FOR I = S8 TO SG - 1:P =  PEEK (I): IF P <  > 32 AND P <  > 76 THEN
    190
170 P =  PEEK (I + 2): IF P < 248 THEN 190
180  POKE I + 2,P - 176: PRINT I
190  NEXT
200  PRINT "MAKING A FEW OTHER ADDRESS CHANGES"
210  POKE SA + 235,74: POKE SE + 168,77: POKE SF + 127,79: POKE SF + 191,
    78: POKE SF + 195,79: POKE SF + 207,237
220  POKE SF + 210,236: POKE SF + 233,195: POKE SF + 253,79: POKE SF + 25
    5,74: POKE SF + 106,163
230  PRINT "READING NEW DATA AND POKING"
240  FOR I = SA + 64 TO SA + 214: READ P: POKE I,P: NEXT
250  FOR I = SA + 253 TO SB + 29: READ P: POKE I,P: NEXT
260  FOR I = SE + 194 TO SE + 201: READ P: POKE I,P: NEXT
270  PRINT "NEWMON IS NOW READY": PRINT : PRINT "TO ENTER IT, CALL 20329"
    : PRINT "OR"
280  PRINT "ENTER MONITOR WITH CALL -151": PRINT "AND THEN KEY '4F69G'"
290  INPUT "SAVE SOFTMON ? ";Q$: IF Q$ >  = "Y" THEN  PRINT "SAVING SOFTM
    ON,A$4800,L$800": PRINT D$"BSAVE SOFTMON,A$4800,L$800"
300  END
310  DATA  255,255,255,32,208,72,104,133,44,104,133,45,162,8,189,16
```

```
320  DATA  75,149,60,202,208,248,161,58,
   240,66,164,47,201,32,240,89
330  DATA  201,96,240,69,201,76,240,92,
   201,108,240,89,201,64,240,53
340  DATA  41,31,73,20,201,4,240,2,177,
   58,153,60,0,136,16,248
350  DATA  32,63,79,76,60,0,133,69,104,
   72,10,10,10,48,3,108
360  DATA  254,3,40,32,76,79,104,133,58,
   104,133,59,32,130,72,32
370  DATA  218,74,76,101,79,24,104,133,
   72,104,133,58,104,133,59,165
380  DATA  47,32,86,73,132,59,24,144,20,
   24,32,84,73,170,152,72
390  DATA  138,72,160,2,24,177,58,170,
   136,177,58,134,59,133,58,176
400  DATA  243,165,45,72,165,44,72
410  DATA  24,160,1,177,58,32,86,73,133,
   58,152,56,176,162,32,74
420  DATA  79,56,176,158,234,234,76,11,
   75,76,253,74,193,216,217,208
430  DATA  211,198,52,32,117,78,76,67,74
```

©

TERMINALL

HAM RADIO INTERFACE

apple & TRS-80

TERMINALL now available for APPLE II ! TERMINALL T2 is a hardware and software system which converts your Apple II computer into a state of the art communications terminal. Simply insert the interface card into any available card slot (1-7) in your APPLE, then plug the attached cables into your radio.

Simplicity. TERMINALL was designed from the outset to be easy to connect to your radio and easy to use. Plug into your receiver headphone jack and copy Morse Code or radioteletype (RTTY). Plug into your CW key jack and send Morse Code. Attach a microphone connector and send Baudot or ASCII RTTY using audio tones (AFSK). That's all there is to hooking it up.

Complete software on DOS 3.2 diskette (Muffin to 3.3), assembled and tested hardware, and extensive instruction manual. Requires an APPLE II or APPLE II Plus, 48K and disk.

The software is loaded into your computer from disk. Enter your callsign and the time and you will start receiving immediately. No settings or adjustments are necessary to receive Morse Code, it's fully automatic - and it works! You may type your message while receiving or transmitting. You will be on the air, receiving and transmitting in any mode, in minutes. As we said, TERMINALL is simple.

■ TERMINALL has the RTTY terminal unit (demod and AFSK - built in. This results in a lower total cost.

■ **Fantastic Morse reception.** Six stage active filter demodulator copies the weak ones. Auto adaptive Morse algorithm copies the sloppy ones. Received code speed displayed on status line.

■ **Outstanding documentation.** Professionally written, 90 page user manual contains step-by-step instructions.

■ **Built in, seperate, multi-stage active filter RTTY and CW demodulators.** No phase lock loops. RTTY demodulator has 170 and either 425 or 850 Hz shift - keyboard selectable - and uses either the panel meter or scope outputs for easy tuning. Copy the weak ones. Copy the noisy ones. Copy the fading ones.

■ **Built in crystal controlled AFSK.** Rock stable for even the most demanding VHF or HF applications. A must on many VHF RTTY repeaters.

■ **Built in 110 or 220** volt AC power supply.

■ **Built in parallel printer driver software.** Simply attach a parallel ASCII printer (e.g. the EPSON MX-80) to your printer port to obtain hard copy in all modes.

■ **Multi level displays** - allows examining and editing of historical text. Word wrapping, word mode editing, diddle, ignore carriage returns, user programmable end of line sequence, adjustable carriage width, multi user - defined WRU, transmit delay (fixed, none or auto adaptive), break mode and more!

■ **The all-in-one TERMINALL** design makes it great for use on HF or VHF Ham, Commercial, SWL or MARS! SWL's: TERMINALL may be jumpered for either 425 or 850 Hz reception to copy news and weather services.

TO ORDER
(209) 667-2888
(209) 634-8888

MACROTRONICS, inc..
C/O DEPT CP
1125 N. Golden State Blvd.
Turlock, California 95380

EDITRIX GRAPHTRIX

EDITRIX + GRAPHTRIX = THE MOST POWERFUL WORD PROCESSOR THIS SIDE OF A NEWSPAPER COMPOSITION ROOM

EDITRIX™
TEXT EDITOR

EASY TO USE
- HELP! Key
- Friendly, COMPLETE instructions that you or your secretary can understand.
- Easy to remember 1 or 2 keystroke commands.
- See your document formatted on the screen AS YOU EDIT IT.

POWERFUL
- 250 Column Horizontal Scrolling.
- Automatic Graphic Insertion and Formatting.
- Automatic Footnote Insertion.
- Underline - Superscript - Subscript - Search - Replace - Block Move.
- Full Printsize, Emphasis, Justify, Margin and Cursor Control.

FLEXIBLE
- Capital letters with ESC or Shift Key modification.
- To be supported by Data Transforms new headline generator coming soon.
- Printout through GRAPHTRIX to 11 different Printers WITHOUT CHANGING YOUR TEXT FILE!

REQUIRES: Apple II with 48K, Applesoft in ROM, DOS 3.3 and the GRAPHTRIX Matrix Graphics System

GRAPHTRIX™
TEXT PRINTER AND GRAPHICS SCREEN DUMP

EASY TO USE
- Complete READABLE documentation.
- Fully Menu Driven.
- Self-running Introduction and Demonstration.

POWERFUL
- Graphic Magnification, Normal/Inverse, Page Centering, Hi and Low Crop Marks, Title String.
- Automatic Formatting of Graphics in your Document.
- Print Size, Emphasis, Underline, Superscript, Footnotes, Chapters, controlled from your text file.

FLEXIBLE
- Prints ANY HI-RES Graphic your Apple II can create.
- Formats Text files from Applewriter OR EDITRIX.
- Use as a Menu Driven Screen Dump OR from in YOUR OWN Applesoft Program.
- Compatible with 11 different Matrix Line Printers AND 7 different Parallel Interface Cards.

REQUIRES: Apple II with 48K, Applesoft in ROM, DOS 3.3 and one of the following line printers: EPSON MX-70/MX-80/MX-100, ANADEX 9500/9501, IDS 440G/445G/460G/560G, CENTRONICS 739, MPI 88G SILENTYPE.

FROM DATA TRANSFORMS, INC., THE GRAPHICS LEADER

Extensibility means the ability to extend a computer language by adding new, customized commands. The & symbol has been used to extend Apple's BASIC, but it has drawbacks. Here's an alternative approach.

Modifying Apple's Floating Point BASIC: An & Interpreter Without the &

H. Cem Kaner and John R. Vokey
Department of Psychology
McMaster University
Hamilton, Ontario

Any computer language has some features which are clumsy and lacks others that you wish you had. If the language is generally satisfactory, you can either put up with these drawbacks or, in some cases, you can "patch" the language, adding new commands or changing old ones to meet your requirements. Applesoft, Apple's floating point BASIC, is quite easy to patch, and there is a growing collection of documentation to help the programmer along. A very direct and popular approach to patching the language uses the &.

& Interpreter

The presence of this character in a program forces a jump out of BASIC, and can force a jump to the subroutine you have written to handle a new command. As the number of available patches has grown, a further feature has been added, an ampersand interpreter. In this case, the & forces a jump to a program (the &-interpreter) which first determines which new command is being signalled, and then branches to the subroutine appropriate to that command. The &-interpreter provided by Mottala is a fine example of this kind of program. For us, this ability to interpret a wide range of user-defined commands is a very significant enhancement to Applesoft. Unfortunately, any ampersand interpreter has a drawback – that the & itself.

The problem arises if the new command flagged by the & involves a symbol Applesoft would readily treat as a command without the &. An example of this is Smith's *& GOTO* command (**COMPUTE!**, May, 1981, #12). This modification to GOTO allows the use of labels instead of line numbers in GOTO statements. For example, if X = 1000, then & GOTO X sends program control to line 1000. Forget that &, though, and Applesoft branches to line 0, no matter what X holds.

This quirk of Applesoft (no fault of Smith's) caused us no end of debugging problems, and we set out to find some way to avoid the new & HEADACHE command the Apple seemed determined to send to us. We found two solutions to the problem. first, use of the & should be restricted to flagging commands which would be gibberish to Applesoft in the absence of a preceding &. Forgetting & still causes the program to crash, but it crashes at the appropriate line number, so the error can be easily found and fixed. Second, when Applesoft commands are themselves modified, the modifying subroutine should be executed whenever that command is encountered, without requiring the presence of the & to signal something new. In this article we will show this second solution can be implemented, on all types of Apple II systems.

It is usually relatively easy to change a BASIC command if Applesoft resides in RAM memory. This occurs in one of two ways: either Applesoft has been loaded onto a 16K memory expansion card, or Ramcard, or Applesoft IIa has been loaded, usually from cassette tape, into the Apple's RAM memory. Ramcard Applesoft is identical to the Applesoft stored in ROM on the Apple II PLUS and on Apple's ROM Applesoft Board. Applesoft IIa is different from these, and we will save discussion of it until later.

If you do use Ramcard Applesoft, then the & approach to modification of BASIC commands is unnecessarily slow along with being very space inefficient. Our labs use the Apple Language System, which includes a Ramcard. By modifying the GOTO code directly, we save 60 of the 67 bytes required for Smith's labelled GOTO/GOSUB program, and do everything his program does, except that our version does not suffer from the one minor error in his (and any ROM Applesoft) version of that routine (details later). Unfortunately, our experience with Apple's rendition of PASCAL (which should not be taken as generalizing to other versions of PASCAL), was quite negative. Partially based on this, other labs in the Department in which we work decided to save their money, and did not buy the Language System or any other RAM memory expansion board. Thus, our programs would not run on their machines, and their &-laden programs would not run on ours. To maintain compatibility of systems, while avoiding the headaches inherent in &-flagged command modifications, we were forced to develop an interpreter for ROM Applesoft which works in much the same way as an &-interpreter would, except that it does not require the &.

How The Applesoft Interpreter Works

Rather than storing the literal characters of BASIC commands, and of many other key words and symbols, Applesoft represents these internally in a tokenized format. Each key word is replaced by a number, between $80 and $EA (see the *Applesoft Reference Manual*, p. 121), and can thus be stored in a single byte of memory. This is an extremely space efficient storage system. However, Applesoft now requires an interpreter to decode these tokens, in order to act on the commands, or to evaluate the functions which they represent. All key words are first tokenized, then interpreted, whether the machine is in Immediate Mode, responding to commands as you type them in, or in Deferred Mode, running a program. Our interpreter mimics the Applesoft interpreter. For this reason, and also because we think it would be helpful for anyone writing modifications to Applesoft, we will describe the behavior of the Applesoft interpreter in some detail.

The Flow Of Control

The Applesoft interpreter starts at location $D805. We cannot list this copyrighted routine here, but you can see it for yourself if you jump to the MONITOR (via CALL -151) and then type in "D805L". Having the routine in front of you may clarify the flow of control described below.

This program begins, at $D805, by determining whether the TRACE command is in effect (flagged by the contents of $F2). If so, and if a Deferred Mode program is being run (flagged by contents of $76), then, before each command is executed, the "#" sign is printed, followed by the line number. The first location following this printout is $D81D, which is branched to directly if the printing is not to be done.

At $D81D, the CHRGET routine ($B1-$C8) is called. This subroutine fetches the next character of the program and sets the Zero flag if that character signals an "end of line," that is, if the character is a carriage return or a colon, ":". The routine then calls, via a JSR, the actual interpretation subroutine, which starts at $D828. This returns immediately, via the RTS at $D857, if an end of line is encountered. Otherwise, the program will return from this subroutine later, in a more indirect way. When this subroutine is returned from, the interpreter exits by jumping to a routine called NEWSTT (NEW STaTement) at $D7D2, which will execute the next program statement, falling back into this interpreter in the process.

If CHRGET did not find an end of line, the $D828 subroutine expects to find a command of some sort. If the character fetched by CHRGET is not a token, the interpreter assumes the programmer intended a LET command (eg X = 100) and jumps to the LET subroutine at $DA46. The interpreter determines whether it has a token by subtracting $80, the value of the smallest token, from the Accumulator (A), which holds the character. If A is still positive, we have a token. This token may represent a command ($80 through $BF), or it may be some non-command key word ($C0 through $EQ). Since we have already subtracted $80 from A, we have a command only if A is less than $40 ($40 + $80 = $C0), which is checked by a CMP (compare) instruction. If A is not less than $40, we do not have a command, which is what should be here, so the interpreter jumps to $D846, thence to $DEC9, which produces a "? SYNTAX ERROR".

The Command Table

If we are dealing with a command, the next job of the interpreter is to determine where to go to execute it. There is an address table, beginning at $D000, (Applesoft IIa: $0800) which contains this information. In this table, the starting address of every command, less 1, is stored in order of magnitude of the command's token. Thus the address of END, whose token is $80, is stored first, from $D000 to $D001. FOR's token is next, and the address of the FOR routine, less 1, is stored from $D002 to $D003. Since $80 was subtracted from A, A now stores a number between $00 and $3F. Double this number, by rotating A left, add it to $D000, and you get the location of the two byte address of the command.

The addition is accomplished by indexing $D001 and $D000 with register Y, after Y is loaded with the doubled contents of A. The command's address, less 1, is then pushed onto the stack. When the next RTS is encountered, the program will "return" control to the last address on the stack, after adding 1 to that address. Thus, the next RTS we encounter will force a jump to the correct starting address of the command to be executed. The actual location of the interpreter's RTS is hidden. The final instruction of the interpreter is a JMP, rather than a JSR, to CHRGET, which will fetch the first character following the command. The RTS from CHRGET is the one which takes us to the command itself.

Note that the next address on the stack is the address of the routine which called this interpreter. We have already seen what happens when this one is returned to: the interpretation process stops and the program jumps to NEWSTT. As soon as the RTS at the end of the command we will execute is encountered, the program will effectively branch to NEWSTT.

What Happens When Applesoft Runs Into An &

The & symbol is tokenized in the same way that BASIC commands are tokenized, so, even though & is not a proper BASIC "command" (see the *Applesoft Reference Manual*), the interpreter will treat it as if it were one. The & address in the token lookup table is $03F5 (less 1). That is, the CHRGET routine jumped to by the interpreter will return to location $3F5 when it has fetched the next character following the &. At $3F5, there are three free locations, which typically contain an instruction to jump to some other address, say $0300, where the actual &-interpreter begins. The &-interpreter will then typically examine the contents of the accumulator, which holds the latest character fetched by CHRGET, and will operate on these in much the same manner as the Applesoft interpreter did for the character it got from CHRGET. There are various ways to accomplish this, but the effect will be to eventually find yet another address (the start of the command flagged by the &) and to force a JMP to that address. The return at the end of the subroutine jumped to will be a return back to the Applesoft Interpreter and, thence, to NEWSTT, as described above.

How To Get Rid Of The & in ROM Applesoft

Our goal is to write a command token interpreter which will handle modified Applesoft commands. We are not worried about things flagged by the & which would not otherwise be treated as commands. We could extend the routine below so that it would handle these, but, given the implementation, (i.e. as a patch to CHRGET, which is very frequently called), this would noticeably slow down Applesoft, so we do not recommend it. Further, we could extend the interpreter so that it would handle tokens which are not commands, allowing modification of functions. Again, we do not recommend this. Applesoft provides a USR function with the explicit intention of allowing user defined functions. Our experience with functions is that they are not really modified at all. Rather, they are replaced by something quite different, which the user considers better.

In this case, we feel that the appropriate vehicle for replacement is the one deliberately built into Applesoft, i.e. USR. An &-based alternative, or an alternative using this routine, would be inappropriate as well as quite clumsy. Given these limited objectives, along with a desire (if only to ease our memory burden) to mimic the Applesoft token interpretation approach, the modification to ROM Applesoft in order to allow compatability with RAM Applesoft without requiring ampersands to flag modified commands is surprisingly straightforward.

Nearly all of ROM Applesoft resides in ROM and so cannot be changed. One important routine, CHRGET, does not reside in ROM. CHRGET is loaded into memory whenever Applesoft is initialized (e.g. by the FP command if you have a disk). This is exactly the routine called by the Applesoft interpreter whenever it wants a new command. We modify CHRGET so that it jumps to a routine we call NEWGET, located at $300, which will replace CHRGET. Along with doing everything CHRGET used to do, this routine checks whether the character it fetches is a token in a list of modified command tokens. If so, it executes that command before returning to the Applesoft interpeter.

All properly written Applesoft commands and programs will leave the TeXT PoinTeR (TXTPTR) pointing to an end of line following the command, once the command has been executed. User written commands must conform to this as well. If they do, then, when NEWGET finally returns control to BASIC, it will pass the Applesoft interpreter an "end of line." This forces, as we have seen above, a branch to NEWSTT, which is just what we want in order to avoid confusion in Applesoft over what should be executed next.

Further Notes

Here are a few further notes on the ROM Applesoft token interpreter subroutine listed at the end of this article, which may not otherwise be clear from the discussion above.

1. The CHRGET routine conceptually divides into two subsections. The first increments TXTPTR, to point to the next character. The next section is often called CHRGOT, and this is the routine which actually fetches the contents of the location pointed to by TXTPTR and sets up various internal flags. By loading JMP $300 into the first three bytes of CHRGET, we effectively destroy the 6-byte segment of code which increments TXTPTR, and thus have to repeat this as the first 6 bytes of NEWGET. In the process, we free the three page 0 locations, $B4, $B5, $B6, which follow the JMP $300. These are used as temporary locations by NEWGET. The CHRGOT routine is completely unaffected by the jump, so we can still use it to actually fetch the character and to set the appropriate flags (eg. Zero).

2. CHRGET has no effect on registers X and Y. We will use both in searching the token table. If we do not return the original values of X and Y when we return from NEWGET, we will induce errors in the many routines which call CHRGET and which assume that this call will not affect these registers.

3. Variable FLAG checks whether or not a command is currently being executed. If FLAG is 0, we are not in the midst of handling a command. If FLAG is not 0, then the user has typed in something like GOTO GOTO, which is a syntax error. Since Applesoft syntax is very clear on this point – all commands must be separated by ends of line (colon or carriage return) – we return "? SYNTAX ERROR" if we find a command token while executing a command. Since BASIC is not in direct control of program execution at this point, this program must do its own error checking.

4. Our lookup table differs in format from the one at $D000. We also store jump locations, less 1, and will get to these via an RTS, as the Applesoft interpreter does. However our table also holds the target tokens themselves. The first and second locations of CMDTBL (CoMmanD TaBLe) hold the low and high bytes of the address of the subroutine which will execute the command whose token is stored in the third location. Similarly for the fourth and fifth locations (addresses) and the sixth location (token), and so on. The command table is stored in this version of the program at location $0354. There is no need whatever to store it here. You can put it in any convenient place in memory, as long as you change the four places in the program which refer to the starting address of CMDTBL (before JMPGOT and after GOTONE).

5. End of CMDTBL is indicated by a zero value for the target token. We load the token into register Y at $31D in order to test for end of table. If Y is zero, the command token is not one we are looking for, so we exit.

The Labelled GOTO/GOSUB Example

1. Comments on the patch.

We use Smith's routine as an example partially because it was published in **COMPUTE!**, so you may be familiar with it, and partially because we have found it quite useful. The routine has been completely rewritten in three ways, once for ROM Applesoft, once for RAMCARD Applesoft, and once for Applesoft IIa, or TAPERAM. A more complete discussion of the logic of the routine is in Smith's article.

The effect of the patch is as follows: Taking X to mean any arithmetic expression or variable, (X may, but need not, be a literal number), then if the value of X is 1000, GOTO X will be treated by Applesoft in the same way as GOTO 1000. Similarly, GOSUB X will be treated as GOSUB 1000. Thus labels can be defined for subroutines (as all rational programming languages, including nearly all assemblers, allow) and for GOTO statements (reminiscent of FORTRAN's ASSIGN statement). If X is a *real* number, it is rounded down to the nearest integer, so be wary of arithmetic expressions.

This patch does not affect the behavior of ON...GOTO and ON...GOSUB, so these must still use line numbers rather than labels. However, you can replace these computed GOTO's in your code by computed GOTO's of a very different type, which are reminiscent of PASCAL's CASE handling. As an example, at the start of your program you might DIMension a matrix SELECT(20) and assign the values of 20 different line numbers to the values of SELECT(I). To GOTO these lines, you can compute the value of I, and then GOTO SELECT(I). With decent commenting on the intention and conditions of each choice of SELECT's line numbers (which is best done at the place in the program that the line numbers are actually assigned to SELECT's elements,) your program will probably be much more readable than one with an equally commented ON...GOTO statement. This is our experience with these two different forms of computed GOTO and GOSUB, and we have stopped using ON...GOTO completely.

Spaghetti Structure

A different method of implementing a computed GOTO is ideal for making your program structure resemble a plate of spaghetti. If you change the value of X at various points in the program and repeatedly use X in GOTO X or GOSUB X statements, then, if you succeed in debugging your program, you will be able to amaze your friends and neighbors with your ability to produce unintelligible, yet functional, code. The labelled GOTO/GOSUB facility as presented here can be used to dramatically increase the readability of your program, or it can be abused to degrade the structure of your program. We strongly recommend that you assign line numbers to specific variables at the start of the program, use informative names for those variables, and never change their values once assigned.

2. The actual patches

The ROM Applesoft patch is given in the appended listing, directly after the lookup table. The program is virtually the same as Smith's, with a few more comments. Note that whether you use this program or Smith's, IF...THEN X, IF...GOTO X and IF...THEN GOTO X will not work properly. We have tried and tried, but cannot fix this flaw in a program of reasonable length. A statement of the form IF...THEN: GOTO X (or GOSUB X) will work correctly. There must be a colon between the THEN and the GOTO or the GOSUB.

The RAM Applesoft patch is much simpler. For either RAM Applesoft version, you do not need the subinterpreter. Instead, modify GOTO directly. GOTO is a subroutine of GOSUB, so this modifies both. Change the JSR $DA0C (the LINe number GETting subroutine) at $D93E (Ramcard), or the equivalent command at $1140 for Taperam to JSR $300 (or wherever you wish to store this patch). This changes the first command of GOTO, forcing it to treat the patch as its line number getting subroutine. The patch consists of:

```
JSR FRMEVL
JSR GETADR
RTS.
```

The Ramcard locations of these routines are $DD7B for FRMEVL, which evaluates the expression following the GOTO or GOSUB and deposits this in Applesoft's floating point accumulator, FAC, and $E752 for GETADR, which moves the contents of FAC to locations $50 and $51, where GOTO expects to find the line number to which to go. The locations for TAPERAM are $157E, and $1F49 for the two routines. That's all there is to it. Seven bytes of new code and, to top it off, IF... GOTO X works (IF...THEN X will not), as do IF...THEN GOTO X and IF...THEN GOSUB X though, to preserve program portability, you will probably want to include the colon, entering only IF...THEN: GOTO (or GOSUB) statements when using labels rather than literal line numbers.

To use this routine, the Applesoft CHRGET routine at $B1 must be modified. This may be accomplished from the monitor by writing B1:4C 00 03NB6:00 or by calling the initialization patch (included below) from BASIC.

```
0300              ;EQUATES:
0300
0300              SYNERR =$DEC9          ;GENERATE SYNTAX ERROR
0300              TXTPTR =$B8
0300              CHRGOT =$B7
0300              XTEMP  =$B4
0300              YTEMP  =$B5
0300              FLAG   =$B6            ;FLAG COMMAND IN PROGRESS
0300
0300              ;THE NEW CHRGET ROUTINE:
0300
0300 E6B8         NEWGET INC TXTPTR      ;INCREMENT LOW BYTE
0302 D002                BNE GETCHR     ;GET NEXT CHARACTER
0304 E6B9                INC TXTPTR+1   ;INCREMENT HIGH BYTE
0306 20B700 GETCHR JSR CHRGOT            ;GET CHARACTER
0309
0309              ;THE SUB-INTERPRETER:
0309
0309 C9C0                CMP #$C0       ;IS IT A NON-COMMAND TOKEN?
030B B025                BCS OUT        ;YES, GO
030D 86B4                STX XTEMP      ;ELSE, SAVE X
030F 84B5                STY YTEMP      ;AND Y
0311 A8                  TAY            ;TEST N FLAG
0312 1016                BPL JMPGOT     ;NOT A TOKEN?, GO
0314 A4B6                LDY FLAG       ;COMMAND IN PROGRESS?
0316 D01D                BNE ERROUT     ;YES, ERROR
0318 A2FF                LDX #$FF       ;NO, SET X FOR TABLE LOOKUP
031A E8           NXTCMD INX             ;POINT X AT NEXT COMMAND TOKEN
031B E8                  INX
031C E8                  INX
031D BC5403              LDY CMDTBL,X   ;GET TOKEN
0320 F008                BEQ JMPGOT     ;END OF TABLE? GO
0322 DD5403              CMP CMDTBL,X   ;ELSE, COMPARE TO TABLE TOKEN
0325 D0F3                BNE NXTCMD     ;NO MATCH, GET NEXT COMMAND
0327 203803              JSR GOTONE     ;ELSE SET UP FORCED JUMP
032A A200         JMPGOT LDX #0
032C 86B6                STX FLAG       ;CLEAR FLAG
032E A6B4                LDX XTEMP      ;RECOVER X
0330 A4B5                LDY YTEMP      ;RECOVER Y
```

```
0332 4CB700 OUT     JMP CHRGOT      ;GET CHAR AT TXTPTR
0335                ;                AND RETURN TO APPLESOFT
0335
0335 4CC9DE ERROUT JMP SYNERR       ;DO SYNTAX ERROR
0338
0338                ;SET UP FORCED JMP VIA RTS AND SET FLAG
0338
0338 85B6   GOTONE STA FLAG         ;FLAG COMMAND IN PROGRESS
033A BD5303        LDA CMDTBL-1,X ;GET HIGH BYTE
033D 48            PHA             ;AND DEPOSIT ON STACK
033E BD5203        LDA CMDTBL-2,X ;GET LOW BYTE
0341 48            PHA             ;AND DEPOSIT ON STACK
0342 60            RTS             ;EXECUTE USER PATCH
0343
0343                ;INITIALIZATION PATCH:
0343                ;A CALL 835 FROM BASIC WILL INITIALIZE
0343                ;THE ROM SUB-INTERPRETER.
0343
0343 A94C   INIT   LDA #$4C         ;LOAD A 'JMP' AND
0345 85B1          STA $B1         ;STORE AT CHRGET
0347 A900          LDA #<NEWGET    ;LOW BYTE OF INTERPRETER
0349 85B2          STA $B2
034B A903          LDA #>NEWGET    ;HIGH BYTE
034D 85B3          STA $B3
034F A900          LDA #0          ;CLEAR THE
0351 85B6          STA FLAG        ;COMMAND IN PROGESS FLAG.
0353 60            RTS             ;RETURN TO BASIC
0354
0354                ;COMMAND TABLE:
0354                ;COMMANDS AND THEIR PATCH ADDRESSES
0354                ;ARE STORED TOGETHER IN LOW-BYTE, HIGH BYTE,
0354                ;COMMAND TOKEN ORDER.  THE LAST THREE BYTES
0354                ;OF THE TABLE MUST BE ZEROS.
0354
0354 5C03   CMDTBL .WOR GOSUB-1     ;GOSUB PATCH ADDRESS
0356 B0            .BYT $B0        ;GOSUB TOKEN
0357 7603          .WOR GOTO-1     ;GOTO PATCH ADDRESS
0359 AB            .BYT $AB        ;GOTO TOKEN
035A 00            .BYT 0,0,0      ;END OF TABLE
035B 00
035C 00
035D
035D                ;AS AN EXAMPLE OF USING THE ROM TOKEN INTERPRETER,
035D                ;WE INCLUDE A LISTING OF A PATCH THAT PROVIDES
035D                ;LABELLED GOSUBS AND GOTOS IN APPLESOFT BASIC.
035D                ;THIS CODE IS FROM M.R. SMITH (COMPUTE, 12, 1981).
035D                ;FOR THE GOSUB PATCH, IT IS EFFECTIVELY A
035D                ;RELOCATION OF THE INITIAL PORTION OF THE
035D                ;APPLESOFT GOSUB CODE.  THIS ENABLES A
035D                ;MODIFICATION OF THE SECTION OF CODE THAT
035D                ;JUMPS TO THE APPLESOFT GOTO CODE, WHERE
035D                ;THE EFFECTIVE CHANGE IS MADE.
035D
035D                ;APPLESOFT POINTERS AND ROUTINES:
035D        CURLIN =$75             ;CURRENT LINE NUMBER
035D        NGOSUB =$D7D2           ;NORMAL GOSUB
```

```
035D            NGOTO   =$D941
    ;NORMAL GOTO
035D            FRMEVL  =$DD7B
    ;EVALUATE EXPRESSION AT TXTPTR
035D            STACK   =$D3D6
    ;CHECK ON STACK POINTER
035D            GETADR  =$E752
    ;TRANSFER FAC TO LINNUM
035D
035D            ;GOSUB PATCH:
035D
035D A903       GOSUB   LDA #3
    ;NORMAL GOSUB RELOCATED
035F 20D6D3             JSR STACK
    ;FROM $D921
0362 A5B9               LDA TXTPTR+1
    ;STORE TXTPTR
0364 48                 PHA
    ;ON STACK
0365 A5B8               LDA TXTPTR
0367 48                 PHA
0368 A576               LDA CURLIN+1
    ;STORE CURRENT LINE NUMBER
036A 48                 PHA
    ;ON STACK
036B A575               LDA CURLIN
036D 48                 PHA
036E A9B0               LDA #$B0
    ;MARK GOSUB
0370 48                 PHA
    ;ON STACK
0371 207703             JSR GOTO
    ;DO A MODIFIED GOTO
0374 4CD2D7             JMP NGOSUB
    ;FINISH NORMAL GOSUB
0377
0377            ;GOTO PATCH:
0377            ;THIS IS WHERE THE EFFECTIVE CHANGE OCCURS.
0377            ;BY REDIRECTING THE EVALUATION OF WHAT FOLLOWS
0377            ;THE GOTO OR GOSUB TOKEN TO FRMEVL (WHICH
0377            ;EVALUATES THE EXPRESSION FOLLOWING THE TOKEN),
0377            ;AND THEN TRANSFERING THE RESULT FROM FAC
0377            ;(THE FLOATING POINT ACCUMULATOR) TO LINNUM
0377            ;(WHERE THE REMAINDER OF THE ACTUAL GOTO
0377            ;ROUTINE EXPECTS TO FIND THE LINE NUMBER),
0377            ;LABEL AND EXPRESSION EVALUATION FOLLOWING
0377            ;GOTO OR GOSUB IS EFFECTED.
0377
0377 200003 GOTO    JSR NEWGET      ;GET NEXT CHARACTER
037A 207BDD         JSR FRMEVL     ;EVALUATE EXPRESSION
037D 2052E7         JSR GETADR     ;TRANSFER FAC TO LINNUM
0380 4C41D9         JMP NGOTO      ;FINISH NORMAL GOTO
0383
0383                .END           ;END ASSEMBLY
```

©

...PET/CBM/VIC? SEE SKYLES...

PET owners everywhere sing

♪♪ Thanks for the Memories ♪♪

to good old Bob Skyles

. . . they should . . . because Bob Skyles is the only complete source for memory boards for *any* PET ever sold. Old Bob won't forget you.

And the Skyles memory systems have the highest quality control of any computer product ever. Over 100 million bits of Skyles memory boards are already in the field. First quality static and dynamic RAMS, solid soldered on first quality glass epoxy. That is why they are **guaranteed**—*in spite of the new lower prices*—**for a full two years.**

The boards, inside the PET/CBM, install in minutes without special tools or equipment . . . just a screwdriver.

Because of our new dynamic memory design, and to celebrate old Bob's 30_{16}^{th} birthday, here are the smashing new prices:

8K Memory System	*orig. $250.00*	*now* **$200.00**	*Save $ 50.00*
16K Memory System	*orig. $450.00*	*now* **$300.00**	*Save $150.00*
24K Memory System	*orig. $650.00*	*now* **$400.00**	*Save $250.00*

. . . *For any PET ever made.* When ordering, just describe your PET by model number and indicate the amount and type (or brand) of memory currently in the unit.

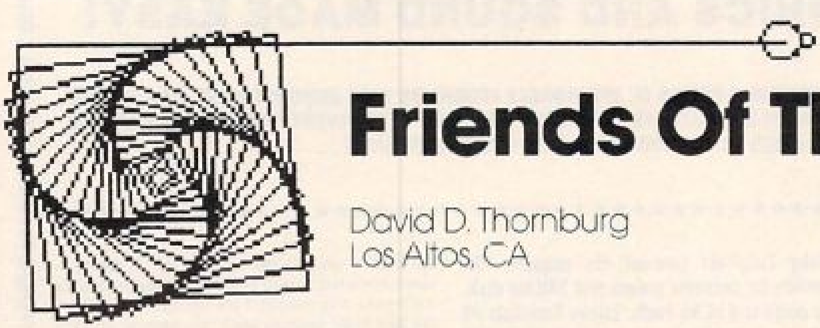
Shipping and Handling . . . . . (USA/Canada) **$3.50** *(Europe/Asia)* **$15.00**

California residents must add 6%/6½% sales tax, as required.

Visa/Mastercard orders: call tollfree **(800) 227-9998** (except California). California orders: please call **(415) 965-1735.**

Skyles Electric Works
231E South Whisman Road
Mountain View, California 94041
(415) 965-1735

...PET/CBM/VIC? SEE SKYLES...



Friends Of The Turtle

David D. Thornburg
Los Altos, CA

The Computer Faire Of The Turtle...

While some might argue with the exact date, I place the start of the Personal Computer Revolution in 1978. That was the year in which affordable desk-top computers were first made available to the general public. The big sellers that year, Commodore, Apple, and Radio Shack, are still going strong – as are other companies who joined in the explosion of enthusiasm which greeted these products.

But during these past years there was another revolution brewing – a revolution in computer languages which promised to make the newly affordable computer easy to program by its largely non-technical owners.

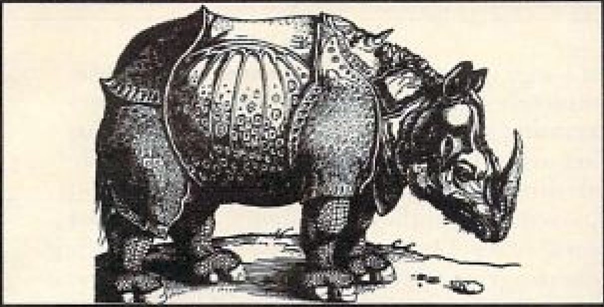
The mainstay of the personal computer revolution was the language BASIC. The fact that many hundreds of thousands of people are able to write programs in this language is strong testimony to its effectiveness. But BASIC has two problems. First, the threshold for learning the language is not very low and, second, the power of the language isn't large enough to invite the user to create extremely sophisticated programs. When BASIC was the only language in town, it was gladly accepted. After all, one alternative – assembly language – doesn't appeal to many first-time computer users; and more powerful structured languages such as PASCAL seem too complex for people interested in balancing checkbooks or generating games.

But, for a decade before 1978, research in university and industrial laboratories was pointing the way to a new type of computer language – a language with a low threshold for learning and a power so great that it could continue to serve the needs of its user at any level of sophistication. One such language, LOGO, was developed and studied on the East Coast, primarily at MIT. While research showed that this language was easy for children to learn and powerful enough for advanced applications, one problem remained – LOGO needed a lot of memory in which to run. As a result, most potential users had to be content either with reading articles about the language or, more recently, with reading Seymour Papert's book, *Mindstorms*.

And then, last year, the seeds of the new revolution began to sprout. Atari released its version of PILOT – a powerful yet simple text manipulation language which had been enhanced by the addition of a graphics environment similar to that in LOGO. At about the same time, Texas Instruments released a version of LOGO which had been compressed to fit on a memory expanded TI 99/4. With these two products, it was evident that a new class of computer language was starting to appear on small affordable computers.

Increasing Literacy

The acceleration of this trend was most evident at the 7th West Coast Computer Faire held in San Francisco this March. This show was packed by attendees who, in my estimation, were the most computer literate group to ever attend this show. In past years an exhibitor was likely to hear questions such as: "Why can't I receive television signals on a color monitor?" This year I was asked questions such as: "What are the major differences between Atari PILOT and LOGO?" Many people had read Papert's book and were fully prepared for the revolution in user friendly languages. They were not let down. The presence of Seymour Papert as keynote speaker and the booths providing information on YPLA (Young People's LOGO Association), FOLLLK (Friends of LISP, LOGO, and Logic for Kids), and FOTT (Friends of the Turtle) set the tone for the release of several versions of LOGO for the Apple II. A special exhibit on the fourth floor of the Faire devoted considerable space to the demonstration of Apple's own LOGO product which was developed by LOGO Computer Systems, Inc. (LCSI). In addition to the language, other exhibits included a prototype of a "sprite" board for the Apple which allows the computer to control four animated turtles at once. Two floors down, another version of Apple LOGO was being offered by Terrapin – a company known previously for its computer controlled robot turtles. (I have copies of both the LCSI and Terrapin LOGO, and



Be sure to enter Krell's giant

$30,000

EDUCATIONAL SOFTWARE CONTEST

WRITE FOR DETAILS. LUDWIG BRAUN. D.E.E.
DIRECTOR OF EDUCATIONAL PRODUCT DEVELOPMENT

College Board SAT* Prep Series

TRS-80, APPLE, PET, ATARI, CP/M, PDP-11 Based on past exams, presents material on the same level of difficulty and in the same form used in the S.A.T.

S.A.T., P.S.A.T., N.M.S.Q.T. — Educator Edition set includes 25 programs covering Vocabulary, Word Relationships, Reading comprehension, Sentence Completion, and Mathematics. $229.95
INDEPENDENT TESTS OF S.A.T. SERIES PERFORMANCE SHOW A MEAN TOTAL INCREASE OF 70 POINTS IN STUDENTS' SCORES.

GRADUATE RECORD EXAM Series — Educator Edition includes 28 programs covering Vocabulary, Word Relationships, Reading Comprehension, Sentence Completion, Mathematics, Analytical Reasoning and Logical Diagrams. $289.95

Micro-Deutsch

MICRO-DEUTSCH set includes 24 grammar lessons, covering all materials of an introductory German course. Four test units also included. Grammar lessons use substitution transformation drills, item ordering, translations and verb drills. Drill vocabulary based on frequency lists. Suitable for use with any high school or college textbook. Extensively field tested at SUNY, Stony Brook. APPLE*, PET*. $179.95

Isaac Newton + F.G. Newton

Perhaps the most fascinating and valuable educational game ever devised - ISAAC NEWTON challenges the players to assemble evidence and discern the underlying "Laws of Nature" that have produced this evidence. Players propose experiments to determine if new data conform to the "Laws of Nature". FULL GRAPHICS NEWTON - presents all data in graphic form. This game is suitable for children. Players may select difficulty levels challenging to the most skilled adults. Both Games $49.95

Pythagoras and the Dragon

Mathematics in a fantasy game context. Based on THE SWORD OF ZEDEK introduces Pythagoras as a mentor to the player. When called on for aid, Pythagoras poses math questions and depending on the speed and accuracy of response, confers secret information. With Pythagoras as an ally, the quest to overthrow Ra, The Master of Evil, assumes a new dimention of complexity. Depending on the level chosen, problems range from arithmetic through plane geometry.
APPLE, TRS80, PET, ATARI* 32K $39.95

ALSO AVAILABLE:

Time Traveler
Odyssey in Time
Sword of Zedek
Krell Game Pak
Super Star &
All Time Super Star Baseball

LOGO

©M.I.T.

Krell's LOGO for APPLE II* Includes: our new low price and

1. Two copies of Krell's LOGO for Apple II*

2. A UTILITY disk containing M.I.T.'s extremely valuable demonstration material, a series of pre-defined shapes which may be substituted for the LOGO TURTLE, and Krell's timesaving library of pre-defined procedures.
3. ALICE IN LOGOLAND, a twenty program tutorial series for intermediate and experienced programmers new to the LOGO language.
4. The technical manual LOGO for Apple II* by H. Abelson & L. Klotz. This is the official technical manual issued by M.I.T.
5. LOGO FOR THE APPLE II* by Harold Abelson, Byte Books. A complete instructional manual for intermediate and advanced users of LOGO.
6. THE ALICE IN LOGOLAND PRIMER, a step by step, lavishly illustrated introduction to LOGO for those who are new to both programming and to LOGO. (available in May and sent free to all purchasers of Krell's LOGO for Apple II*).
7. A comprehensive wall chart that portrays, explains and graphically illustrates the LOGO commands in action.
8. A one-year's free subscription to the LOGO & EDUCATIONAL COMPUTING NEWSLETTER. (Regularly $30.00 per year). A new authoritative source of information about the structure, classroom application and capabilities of LOGO, plus an up to the minute forum on the most significant new ideas and issues in educational computing.

$149.95

Competency/Proficiency Assessment Examination Preparation Series

A COMPREHENSIVE MULTI-MODULE TUTORIAL AND EXAMINATION SIMULATION PACKAGE

- Diagnostic analysis
- Prescription of individual study plans
- Systematic instruction in all facets of math and verbal skills
- Unlimited drill and practice with selection by skill area or level of difficulty
- Worksheet generation and performance monitoring
- Practice exam simulations
- A complete record management system

Krell's unique logical design provides personalized instruction for each student according to individual needs.

APPLE, ATARI, COMMODORE CBM/PET, CP/M, I.B.M.
RADIO SHACK TRS-80, TEXAS INSTRUMENTS 99/4(A)*
CALIFORNIA, N.Y., NATIONAL EDS. CALL FOR DETAILS & PRICES

Krell Software Corp.

"The State of the Art in Educational Computing"

1320 STONY BROOK ROAD / STONY BROOK NY 11790 / (516) 751-5139

Krell Software Corp. has no official ties with the College Entrance Examination Board or the Educational Testing Service. Krell is, however, a supplier of products to the E.T.S. N.Y.S. residents add sales tax.
PROGRAMS AVAILABLE FOR THE TRS80, APPLE II, PET & ATARI.
*Trademarks of Apple Comp Corp., Tandy Corp., Commodore Corp., Atari Corp.

will report my opinions to you in a later column. At first glance, they are each terrific!)

Just as some people feel that IBM has legitimized the personal computer by entering the market themselves, one got the feeling that the presence of LOGO on the Apple (with its massive installed base) was going to help wean people from BASIC faster than might otherwise be expected.

Product and information booths were only one source of information on this topic. In addition, no less than a dozen demonstrations, workshops, tutorials and speeches were devoted to user friendly languages.

While I spent much of my time helping Addison Wesley show my book on turtle geometry (yes, Atari PILOT fans, *Picture This!* is now at your local bookstore!), I was still able to talk with many attendees and visit the other booths. Those people who knew about LOGO could hardly wait to see a version on their own computer.

Decked out in my Friends of the Turtle T-shirt, I visited with Rich Pattis who was demonstrating software supporting his excellent book, *Karel the Robot* (Wiley) – a book devoted to introducing people to programming through the medium of the turtle. While geared towards the beginning PASCAL programmer, Pattis' work shows a sensitivity that is characteristic of the user friendly languages such as PILOT and LOGO. The people from FOLLLK were acting as guides to the host of LOGO-based exhibits and talks. Larry Muller and his dad, Jim, demonstrated TI LOGO at the YPLA booth. Recent price reductions in both the TI 99/4A computer and in the TI LOGO cartridge have further fanned the flames on a product whose sales were already heating up quite nicely.

The Computer As A Mudpie

Considering the booths, talks, workshops, and enthusiastic attendees, it was clear that this year's Computer Faire was the focal point of the new revolution – the user friendly languages had come home at last. People were lined up against the walls to hear Papert's keynote address. While much of his talk was devoted to describing the function of the World Center for Informatics and the Human Resource (in Paris, France), he also talked about his view of the computer as a "mudpie" – a tool with which children could (through languages such as LOGO) make discoveries on their own and with which they could acquire for themselves information which was previously "taught" to them by teachers. It was easy to be swept along in the belief that we were witnessing the onset of a revolution which promises to be as significant as the advent of the personal computer itself.

I can state, without equivocation, my belief that languages such as LOGO and PILOT will completely displace BASIC as the popular programming medium in the next five years. This belief arises not from my own excitement with something new, but from the results of my own experiences with these languages over the past several years. I have had the pleasure of sharing these programming environments with children from second to sixth grade, as well as with teachers, college students, and artists. The enthusiasm expressed by these varied audiences is enormous.

And each of you who calls yourself a Friend of the Turtle is sharing in this new age of computing – in this new level of power now being unlocked in the Apple, Atari and TI computers all over the world.

Let the revolution continue!

For more information on LOGO and other user friendly languages, contact:

Young People's LOGO Association
1208 Hillsdale Dr.
Richardson, TX 75081

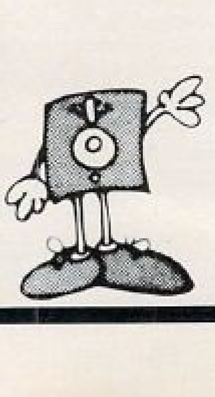
The FOLLLK Foundation
c/o Social and Information Service
HLL 382
San Francisco State University
1600 Holloway Ave.
San Francisco, CA 94132

and, of course,

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

©

COMPUTE!
The Resource.



MEMOREX
FLEXIBLE DISCS

WE WILL NOT BE UNDERSOLD!! Call Free **(800)235-4137** for prices and information. Dealer inquiries invited and C.O.D.'s accepted.

PACIFIC
EXCHANGES
100 Foothill Blvd.
San Luis Obispo, CA
93401. In Cal. call
(800) 592-5935 or
(805) 543-1037

VISA

Here's an alternative to Apple Pascal's Transfer function when you need hardcopy listings.

Apple Pascal Lister

Scott Barrus
Ramsey, NJ

The lister program below is written in modules. The first is self-explanatory. It puts a header on the top of the screen. The second module goes with the first. It is used to delay long enough for a person to read the header. If a shorter or longer time is desired, the number 3000 can be changed.

The module Input puts the prompt "File name" on the left-hand side of the screen, 15 spaces down. It also names a string "namefile" to be used with the names module. If a name is entered without a ".text" on the end, this module attaches it. A possible addition would be to prompt and ask if ".text" were wanted.

The module Names builds and writes the Filename and page # on the top of each sheet. It also then puts 79 "=" 's underneath the filename line. Since Pascal does not have a tab function, either a tab type module would have to be written or, just use spaces to separate the word "page" and file names. If a further right position were wanted, then more blanks could be added or a tab module made up.

Getfile does the hard work. It reads a line from the input file and then writes each line to the printer. It does this 55 times and then sends a form feed to the printer. It will keep doing this until it comes across an end of file (EOF) marker at which time this module ends.

Openfile opens the avenue for the program to work. It opens the text file to be read and it activates the printer as a file named outfile.

Procedure Epson is a bit of customizing because the author uses an MX-80 printer. Since the program can print more than one file at a sitting, the change mode option was added to give the user flexibility. If the printer being used does not have all the capabilities of the Epson, either eliminate the lines that cannot be used or eliminate module Epson. If different codes are used on any printer, those codes can be substituted for the Epson codes.

The final section is the body of the program. The program starts by assigning the value of one to "pagenumbers." A header comes next. The file must be opened so that when we state what Epson code is to be used the printer will be able to get the code. All that is left is to let the program get (and print) out the files. When done, the user is prompted for another file. If none is wanted, the screen is cleared and the user gets a cheerful message. Finally the file opened is closed and the program ends.

To run the program, type X from the main prompt line in Pascal. When asked what file to execute, type in the name that was assigned to the compiled code. To run the program as on the example disk, type in List:Lister.code. The program will be off and running.

```
Program Lister;

  const

     Pagelength = 55;
     Linelength = 79;

   var
     files,
     Pagenumbers,
     time               : Integer;

     Filename,
     Namefile           : String;

     Line,
     Nametitle,
     Separator          : String [ 255 ];

     Infile,
     Outfile            : Text;

     Ans,
     answer,
     choice             : Char;

Procedure Header;
   begin
     page(output);
     Writeln('     *******************************');
     Writeln('     *                             *');
     Writeln('     *     Apple Pascal Lister     *');
     Writeln('     *       by Scott Barrus       *');
     Writeln('     *                             *');
     Writeln('     *******************************');
   end;

Procedure Wait;
  begin
    for time := 1 to 3000 do;
  end;

Procedure Input;
  begin
    Page(output);
     Gotoxy(1,15);
     Write('File name: ');
     Readln(filename);
      Namefile := filename;
        if pos('.',filename)=0 then
          filename := concat ( filename,'.text');
  end

Procedure Names;
  begin
  Write(outfile,chr(14),namefile,chr(20));
  Writeln(outfile,'                          Page ',pagenumbers);
    Separator := '=';
   for files := 1 to linelength do
    Separator := concat(Separator,'=');
  Writeln(outfile,Separator);
  Writeln(outfile);
  end;

Procedure Getfile;
  begin
   Repeat
     Names;
     For files := 1 to Pagelength do
       begin
         Readln(infile,line);
         Writeln(outfile,line);
       end;
     Page(outfile);
     pagenumbers := pagenumbers + 1 ;
   Until EOF(infile) = True;
  end;

Procedure OpenFile;
  begin
    Rewrite(outfile,'printer:');
    Reset(infile,filename);
  end;

Procedure Epson;
  begin
    repeat
    Page(output);
      Writeln('Choose:');
      Writeln;
```

MicroStand™

Standard in Microcomputing Products

AT YOUR LOCAL DEALER OR ORDER DIRECT

DISK CADDY DC-1...why be disorganized when you can be DISKorganized! Let MicroStand's DC-1 solve the problem of practical diskette storage. The DC-1 is constructed of sturdy steel and offers a 100 diskette capacity. Color and texture matched to the Apple, it's a handsome and practical addition to your microcomputer station. Commercial users, ask us about our 8" disk model.

ACCESSORY BRIDGE AB-1...shelves your Epson MX-70/80/100 or Bytewriter printer and keeps paper neatly tucked underneath. Made of sturdy steel and matched to your Apple's color and texture.

DC-1 $49.95 plus $5.00 shipping and handling
AB-1 $39.95 plus $5.00 shipping and handling

MicroStand
P.O. Box 96
Tolovana Park, OR 97145
1 (503) 436-1429

VISA MasterCard

```
    Writeln('(A).......Emphasize');
    Writeln('(B).......Double Strike');
    Writeln('(C).......Both');
    Writeln('(D).......Cnx Either');
    Writeln('(E).......Continue as is');
    Writeln;
    Write('choice: ');
    Readln(choice);
  until choice in [ 'A','a','B','b','C','c','D','d','E','e'];
  case choice of
    'A','a'   : Writeln(outfile,chr(27),'E');
    'B','b'   : Writeln(outfile,chr(27),'B');
    'C','c'   : Writeln(outfile,chr(27),'E',chr(27),'B');
    'D','d'   : begin
              Page(output);
               repeat
                Writeln('Cancel which?');
                Writeln('(A)...Emphase');
                Writeln('(B)...Double Strike');
                Writeln('(C)...Both');
                Writeln('(D)...Continue as is');
                Write('Choice: ');
                Readln(choice);
               until choice in ['A','a','B','b','C','c','D','d'];
               case choice of
                'A','a'  : Writeln(outfile,chr(27),'F');
                'B','b'  : Writeln(outfile,chr(27),'H');
                'C','c'  : Writeln(outfile,chr(27),'F',
                                          chr(27),'H');
                'D','d'  : Exit(Epson);
               end; [ case ]
            end;
      'E','e'   : Exit (Epson);
   end; [ case ]
end; [ epson ]

 Begin [ Main Program ]
    pagenumbers:= 1;
    Header;
     Wait;
    Repeat
       Input;
       Openfile;
         Epson;
       Getfile;
         Write('Another file? (Y/N) ');
         Readln(answer);
   Until answer in ['N','n'];
   Page(output);
   Gotoxy(1,15);
   Writeln('Have a Nice Day!');
   Close (infile);
end.
```
©

COMPUTE! The Resource.

...PET/CBM/VIC? SEE SKYLES...

PET owners everywhere sing

Thanks for the Memories

to good old Bob Skyles

. . . they should . . . because Bob Skyles is the only complete source for memory boards for *any* PET ever sold. Old Bob won't forget you.

And the Skyles memory systems have the highest quality control of any computer product ever. Over 100 million bits of Skyles memory boards are already in the field. First quality static and dynamic RAMS, solid soldered on first quality glass epoxy. That is why they are **guaranteed**—*in spite of the new lower prices*—**for a full two years.**

The boards, inside the PET/CBM, install in minutes without special tools or equipment . . . just a screwdriver.

Because of our new dynamic memory design, and to celebrate old Bob's $30\frac{th}{16}$ birthday, here are the smashing new prices:

8K Memory System	*orig. $250.00*	*now $200.00*	*Save $ 50.00*
16K Memory System	*orig. $450.00*	*now $300.00*	*Save $150.00*
24K Memory System	*orig. $650.00*	*now $400.00*	*Save $250.00*

. . . *For any PET ever made.* When ordering, just describe your PET by model number and indicate the amount and type (or brand) of memory currently in the unit.

Shipping and Handling . . . . . (USA/Canada) **$3.50** *(Europe/Asia)* **$15.00**

California residents must add 6%/6½% sales tax, as required.

Visa/Mastercard orders: call tollfree **(800) 227-9998** (except California). California orders: please call **(415) 965-1735.**

Skyles Electric Works
231E South Whisman Road
Mountain View, California 94041
(415) 965-1735

...PET/CBM/VIC? SEE SKYLES...

A simple game illustrating a most sophisticated topic. The idea of recursive subroutines *is one of the more complex notions in programming – a subroutine which is its own subroutine. "Towers of Brahma," a child's game with pegs and disks, is used to illustrate how recursion works with versions here for PL/I and Microsoft (Commodore, Apple, OSI) and Atari BASICs.*

Recursive BASIC Subroutines

Earl Wuchter
Catasauqua, PA

Recursive *(adj): of, relating to, or constituting a procedure that can repeat itself indefinitely, or until a specified condition has been met.*

Subroutine *(n): a sequence of computer instructions for performing a specified task that can be used repeatedly in a program or in different programs.*

– Webster's New Collegiate Dictionary

Used together, these two words describe one of the most powerful, and most fascinating entities in programming, but it is a rare bird.

Recursive subroutines are most useful in compiled languages, but are seldom allowed. On the other hand, an interpreted language like BASIC can't help but allow a subroutine to reenter itself, but some serious problems must be overcome before the technique is of any use.

The problem with BASIC is that all variables are *global.* (When any variable is changed in a subroutine it is changed for the whole program.) To see what is missing in BASIC, look at a *call* statement and a subroutine statement in a typical compiled language. The variables in parentheses are called arguments or parameters:

```
CALL TRY (A,X)
SUBROUTINE TRY (A,Z)
```

When TRY is invoked, A and Z will take on the values of A and X from the calling routine. On return, A and X will receive the values of A and Z. Even though both routines use the variable A, the A in the calling routine will not be changed until a return is executed. The status (global or local) of other variables is normally up to the programmer.

BASIC subroutines do not have arguments. The normal programming practice is to have the calling routine assign values to subroutine variables before the GOSUB is executed. To prevent a subroutine from inadvertently affecting any other routine, we give it a set of local variables by using names that do not appear in any other routine.

A Solution to Passing BASIC Variables

Now you can see the problem with recursive code. How can we give a subroutine a set of variables that are different from those in the calling routine when they are one and the same routine?

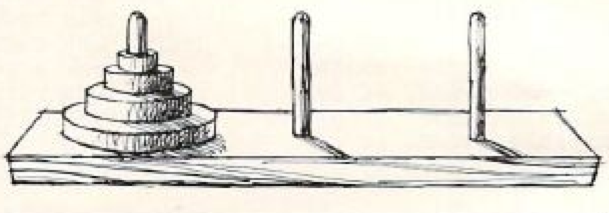
There is a solution. We can create an array of arguments and have each generation of the subroutine use its own "row" in the array.

The Towers Of Brahma

The "Towers of Brahma" is one problem that is best solved with a recursive subroutine. The solution, in turn, best illustrates the capability of recursive code. The two seem to have been made for each other.

This interesting mathematical problem is often found disguised as a child's toy: a board with three pegs and four or five disks of varying diameters that fit on the pegs. The disks are initially arranged on the first peg in order of their diameters, with the smallest one on top. They are to be moved one at a time, always keeping the smaller ones on top, until they are on the third peg.

Towers of Brahma



When the recursive solution to this problem is coded in the proper language, it is a thing of beauty. A PL/I version of the subroutine is shown below. This will be the model for a BASIC version. The main routine is not shown. The main routine simply inputs the number of disks and makes the initial call:

```
CALL MOVE (NDISKS,1,3);
```

Note: In this version of PL/I, arguments passed in as expressions become dummy arguments and do not change anything in the calling routine. Adding zero to a variable makes it an expression without altering its value. This feature provides for extremely compact code.

```
MOVE: PROCEDURE (N,F,T) RECURSIVE;
IF N=0 THEN RETURN;
CALL MOVE (N-1,F+0,6-F-T);
PUT SKIP DATA (N,F,T);
```

Brief Definitions

Some of the terms used in this article might be unfamiliar. Here's a short description of the main ideas:

CODE – Refers to the statements, the lines, or the entirety of a program, as in the phrase "there's an error in your code at line 110." Also used as a verb, as another word for programming: "I'm coding a subroutine."

COMPILED LANGUAGE – Unlike BASIC (an *interpreted language*) there are three steps to getting a program to run when you're working with a compiled language: 1. Write it. 2. Compile it. 3. Run it. After you write the program, a separate program called a *compiler* translates what you've written into a form of *machine language*. This results in a program which can then be executed (run).

Some examples of compiled languages are FORTRAN, COBOL, and PL/I. The language the program is written in (step 1, above) is called the *source language* and when you finish typing it you've got not surprisingly, the *source program*. Step 2 is where you instruct the compiler to "examine" and transform the source program into an *executable* (it can be executed) program called the *object* or *target* program.

You can write, easily read, and easily modify source programs, but you can't run them. Object programs, on the other hand, do run, but they are difficult to read (they're not written in words like PUT IF THEN DECLARE, but rather in machine or assembly language.)

INTERPRETED LANGUAGE – Step 2 (above) does not take place in an interpreted language. There are only two steps to writing an interpreted language program (BASIC is an example): 1. Write it. 2. RUN it. However, while the program is RUNning, a separate program in the computer is "interpreting" the meaning of words like PRINT and INPUT. In other words, the computer slows down somewhat while the interpreter decides what PRINT means and just what kind of PRINT is involved (to the screen, the printer, a disk file?). Then the interpreter sets up pointers and flags and variables and momentarily sends control to the machine language subroutine (in a library of such subroutines) which can do a PRINT.

All of this translating and interpreting is going on *during* the RUN of the program. It's obvious why interpreted programs tend to RUN more slowly than compiled programs. On the other hand, because you can write it/RUN it, it is easier to test parts of a program immediately and make the necessary changes to the program itself. Debugging is easier simply because the program you write is the same program that you will RUN.

PARAMETER PASSING – This generally refers to the ways that variables are made available to several programs at once, (or, if the variables are local, to several parts of the same program at once). For example, if the variable CASH "holds" 15.98, we might want to "pass" that value (15.98, the parameter) to another program. Some computers will cancel all variables when a new program is LOADed into RAM memory. How can the 15.98 be "passed" to the new program?

Likewise, if variables are local, the value of CASH will be different in subroutines from what is in the main body of the program unless 15.98 is "passed" to the subroutines.

Passing parameters is handled in different ways depending on which computer or language is being used. Compiled languages often allow you to put the variables in parentheses on the same line as a CALL to or RETURN from a subroutine. The example CALL SUB3 (CASH, TOTAL) *passes* the values of CASH and TOTAL to the subroutine SUB3. It can later pass variables back to the main routine in the same way, using parameters (also called *arguments*).

PL/I – A "high level" language (as opposed to lower level, closer to machine language) which combines attributes of FORTRAN and COBOL.

DUMMY ARGUMENTS – Parameters which have no effect other than to take up some necessary space. An example is the FRE(1) statement in Microsoft BASIC where the value within the parentheses can be anything. It's just there because the computer will not recognize FRE().

NESTING – When something is contained by something else. FOR I = 1 TO 10: FOR J = 1 TO 2: NEXT J: NEXT I. This J loop is said to be *nested* within the I loop.

STACK – The computer must be able to remember "return addresses." 100 GOSUB 500 will turn over control to whatever is on line 500, but eventually there will be a RETURN. So, before GOSUBbing, the "address" to RETURN to is put on the computer's *stack* and later pulled off the stack by RETURN. There is a limit to how many return addresses can be pushed on the stack. This is normally no problem since most subroutines go right back via RETURN, relieving the stack of the address number. Recursive, self-calling subroutines, however, aren't RETURNing. It's GOSUB-GOSUB-GOSUB, etc. Unless the recursion is carefully managed, the stack could quickly fill with return addresses and is then said to *overflow*.

CALL MOVE (N-1,6-F-T,T+0);
END MOVE;

The BASIC program has arrays for variables N, F, and T. Variable G serves as the index to the arrays. G represents the number of nested GOSUBs. The subroutine increments G on each entry and decrements it on each return, thereby keeping count of its own generation number. Arguments are passed to the next generation by putting values into the arrays at (G+1) before each GOSUB. The variable G is always the current subscript for any generation of the subroutine.

This method of using arrays requires that you know beforehand what the maximum nesting depth will be in order to dimension the arrays. This number is a function of the problem, but may be limited by the BASIC interpreter.

Solving 21 Disks Would Take More Than Eight Days

The program runs in less than 2K. The internal stack size of my PET restricts the number of disks to 21. When I try to run more than 21 I get an "OUT OF MEMORY" message. Your system may allow more or less, but don't expect to run this many disks to completion. The time required to complete 21 disks (at a rate of approximately three moves per second) will be more than eight days, and one more disk would double that.

The BASIC program here shows the moves to be made, along with a plot of the nesting level made down the left side of the screen. Each increase in the length of the plotted line indicates a GOSUB. Each decrease in length indicates a RETURN. You can see a more detailed view of the process by replacing the plot with a printout of the current array variables.

Program 1. Microsoft Version

```
10 REM TOWERS OF BRAHMA (RECURSIVE)
11 DIM N(22),F(22),T(22)
12 P$="----+----+----+----+--"
13 INPUT"N DISKS (1 TO 21) ";N(1)
14 IFN(1)<1 OR N(1)>22 THEN 13
15 F(1)=1
16 T(1)=3
17 GOSUB 31
18 END ::::::::::::::::::::::::
31 G=G+1
32 PRINT LEFT$(P$,G)
33 IF N(G)=0 THEN 43
34 N(G+1)=N(G)-1
35 F(F+1)=F(F)
36 T(G+1)=6-F(G)-T(G)
37 GOSUB 31
38 PRINT TAB(19)"DISK#"N(G)"FROM"F(
   G)"TO"T(G)
39 N(G+1)=N(G)-1
40 F(G+1)=6-F(G)-T(G)
41 T(F+1)=T(F)
42 GOSUB 31
43 G=G-1
44 PRINT LEFT$(P$,G)
45 RETURN
```

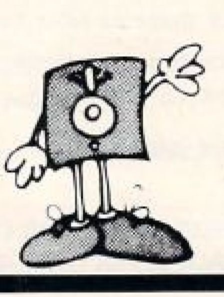
Program 2. Atari Version

```
11 DIM N(22),F(22),T(22),P$(20)
12 P$="----+----+----+----+---"
13 ? "N DISKS (1 TO 21)";:INPUT T:N(1)=T

14 IF T<1 OR T>21 THEN 13
15 F(1)=1
16 T(1)=3
17 GOSUB 31
19 END
31 G=G+1
32 IF G THEN ? P$(1,G)
33 IF N(G)=0 THEN 43
34 N(G+1)=N(G)-1
35 F(G+1)=F(G)
36 T(G+1)=6-F(G)-T(G)
37 GOSUB 31
38 ? ,"DISK #";N(G);" FROM ";F(G);" TO "
;T(G)
39 N(G+1)=N(G)-1
40 F(G+1)=6-F(G)-T(G)
41 T(G+1)=T(G)
42 GOSUB 31
43 G=G-1
44 IF G THEN ? P$(1,G)
45 RETURN
```

©

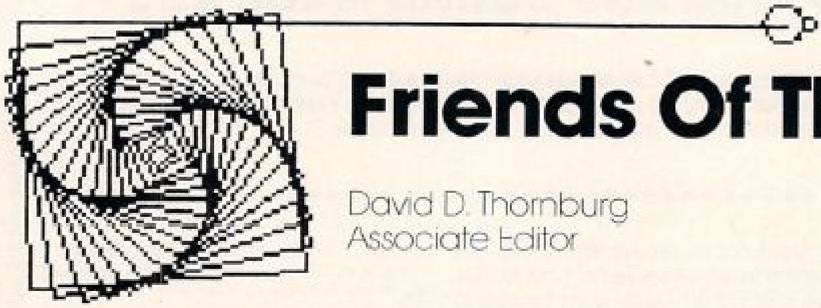
MEMOREX
FLEXIBLE DISCS

WE WILL NOT BE UNDERSOLD!! Call Free **(800)235-4137** for prices and information. Dealer inquiries invited and C.O.D.'s accepted.



PACIFIC
EXCHANGES
100 Foothill Blvd.
San Luis Obispo, CA
93401. In Cal. call
(800) 592-5935 or
(805) 543-1037

VISA



A Monthly Column

Friends Of The Turtle

David D. Thornburg
Associate Editor

A Fractal Of My Former Self...

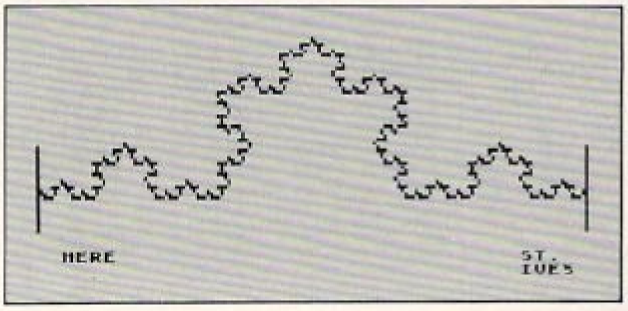
As I was going to St. Ives,
I met a man with seven wives.
Each wife carried seven sacks;
And in each sack was seven cats;
And with each cat was seven kits.
Kits, cats, sacks, and wives,
How many were going to St. Ives?

If you remember this puzzle from your childhood you probably remember that the answer is one.

Well, that may not be very exciting; but suppose you are walking to St. Ives and you want to know how far it is from some other location – Here, for example. The first thing we might do is look at a map.

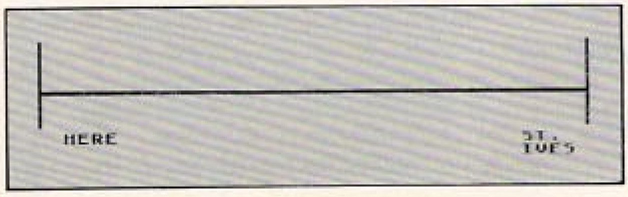
From the map we see that St. Ives and Here are both coast towns located near bays. In between, the coast juts out to sea.

Figure 1.



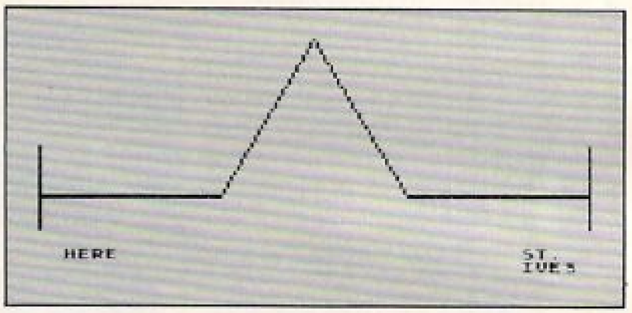
How long is the coastline between the towns of Here and St. Ives? If we set a pair of dividers to span the distance between the two towns we get one measurement. This distance is one kilometer.

Figure 2.



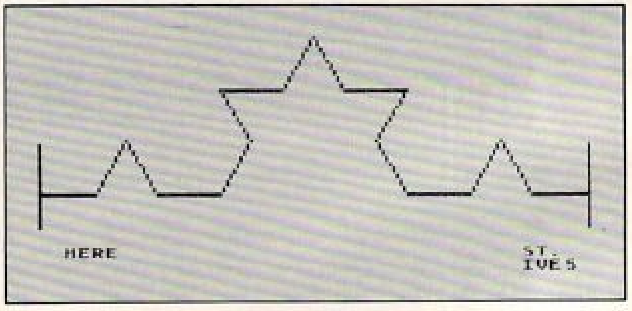
But this path doesn't take us along the coast. If we set our dividers to measure in units of 1/3 of a kilometer, we can trace a path which more closely follows the coastline.

Figure 3.



By counting each 1/3 kilometer increment we can see that this path is 4/3 kilometer in length. Next, let's set the dividers to 1/9 of a kilometer and measure the distance again.

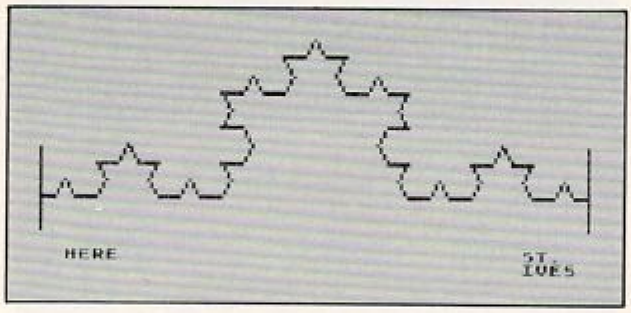
Figure 4.



This looks more like the actual coastline and gives us a distance of 16/9 kilometers. Since we replaced each straight line in Figure 3 with a replica of Figure 3, we can see that the length of the coast in Figure 4 can be written as 4/3 * 4/3.

Now let's set the dividers to 1/27 of a kilometer and measure again.

Figure 5.



This brings us even closer to the coastline and gives us a distance of 4/3 * 4/3 * 4/3 or 64/27 kilometers – more than twice our original estimate.

By now it should be obvious that each time we reduce the setting of our dividers by 1/3, the path length increases by a factor of 4/3. If we were to walk along the coast by following every nook and cranny, the distance would be 4/3 * 4/3 * 4/3 * 4/3 * 4/3,... etc. In other words, the length is infinite.

The reason for this is that each part of the coastline is an infinite set of replicas of the overall shape shown in Figure 3. If you were to look at a high magnification view of the coastline it would be just as bumpy as a lower magnification view.

Real coastlines don't have infinite lengths, of course, but our imaginary coastline does because it is made from a type of self-similar curve called a *fractal*. Mathematical expressions of this type have been known for some time but were not studied much because mathematicians thought their properties were too strange. For an insight into just how strange these functions are, you might want to look at the book *Fractals: Form, Chance, and Dimension* by Benoit Mandelbrot (Freeman 1977). This book discusses the properties of curves such as the one I chose for the imaginary coastline. This function is called a Triadic Koch curve and was discovered by von Koch in the early 1900's.

Mandelbrot's book provides an interesting glimpse of this strange mathematical world, but is quite frustrating in that he doesn't show the reader how to express these curves so that a computer could draw them.

Have you guessed that the turtle might hold the key to drawing fractals?

First, let's look at Figure 3, since this is the basic building block from which all the other curves are made. A procedure for creating this figure is shown below in both Apple LOGO and Atari PILOT. (This also lets you see the similarities and differences between these two languages.)

```
Apple LOGO           Atari PILOT
TO F0   :SIZE        *F0
FORWARD   :SIZE      GR: DRAW #A
LEFT 60              GR: TURN -60
FORWARD   :SIZE      GR: DRAW #A
RIGHT 120            GR: TURN 120
FORWARD   :SIZE      GR: DRAW #A
LEFT 60              GR: TURN -60
FORWARD   :SIZE      GR: DRAW #A
END                  E:
```

If you move the turtle to the left edge of the screen and turn it to face right, then, by using the procedure F0, a replica of the curve in Figure 3 should appear on your display. For a line length of 81 units (a good size for Apple LOGO) you would type F0 81. In Atari PILOT you might want to use a line length of 54 units (because of screen resolution differences between the Atari and Apple graphics). To do this, just type:

```
C: #A=54
U: *F0
```

and you should see a curve similar to that in Figure 3.

Now, what about Figure 4? In our coastline, each straight line section is replaced with a copy of Figure 3 which has been reduced by one-third. Let's call the procedure to do this F1.

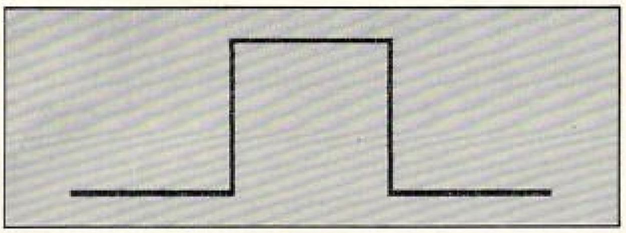
```
Apple LOGO           Atari PILOT
TO F1   :SIZE        *F1
F0   :SIZE/3         U: *F0
LEFT 60              GR: TURN -60
F0   :SIZE/3         U: *F0
RIGHT 120            GR: TURN 120
F0   :SIZE/3         U: *F0
LEFT 60              GR: TURN -60
F0   :SIZE/3         U: *F0
END                  E:
```

Suppose we next want to draw the next level of this curve. All we need to do is create a copy of F1 (called F2) in which references to F0 are replaced by references to F1. Each time this procedure is used it will use F1 which will use F0. If you follow this process a few more times, you might make a procedure called F20 which uses F19, and so on.

How far do we need to go? If our original curve (Figure 3) used a line length of 27, for example, then each line should be 9 units long for F1,3 units long for F2 and 1 unit long for F3. Since the display can't show lines less than 1 dot long, it hardly makes sense for use to try to make this curve with any finer resolution.

In a future column we will show how LOGO lets you create these curves with just one procedure. Meanwhile, you can test your abilities by using a different figure as a starting pattern:

Figure 6.



Because of the 90 degree turns in this figure, those of you with WSFN will also be able to make this fractal curve.

Here is the first procedure you will need:

```
Apple LOGO              Atari PILOT
TO F0   :SIZE           *F0
FORWARD    :SIZE        GR:  DRAW #A
LEFT 90                 GR:  TURN -90
FORWARD    :SIZE        GR:  DRAW #A
RIGHT 90                GR:  TURN 90
FORWARD    :SIZE        GR:  DRAW #A
RIGHT 90                GR:  TURN 90
FORWARD    :SIZE        GR:  DRAW #A
LEFT 90                 GR:  TURN -90
FORWARD    :SIZE        GR:  DRAW #A
END                     E:
```

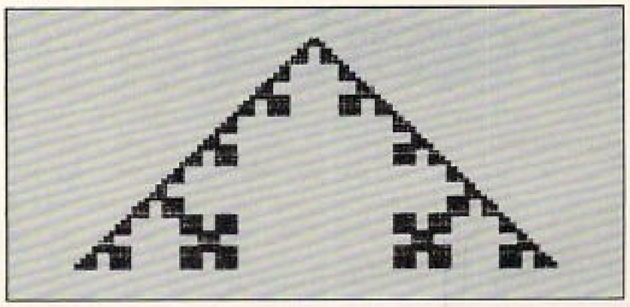
In WSFN you would create the procedure *A this way:

```
=*A(F6RF2RF2RF6RF)
```

(You might want to have the turtle move forward more with each step for the first few levels so you can see what is going on more easily.)

Here is what you should get for your final picture:

Figure 7.



Keep experimenting with other patterns – and let me know how you are doing.

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

©

Good news for kids of all ages . . .

Robby looks great in color.

subLOGIC
Communications Corp.
713 Edgebrook Drive
Champaign, IL 61820
(217) 359-8482
Telex: 206995

SubLOGIC is pleased to announce the release of 4 new programs for the Atari® 800™ computer:

AT-AG1 (arcade game)
ROBBY THE ROBOT CATCHER
Baby robots are falling from the skies. Catch them if you can. 32K

AT-FA1 (fantasy adventure)
ADVENTURE ON A BOAT
Go fishing for points in the waters of Fantasy where anything can happen. 32K

AT-FA2
GHOSTLY MANOR
Find a magic key to defeat the evil spirits, before they can do the same to you. 32K

AT-FA3
THE BLACK FOREST
Explore the Black Forest in search of the life-giving waters of Amrita. And beware of demons along the way. 48K

High Quality, Pure Fun . . .
for anyone who is young at heart. **$24.95.**

See your dealer.

For direct orders add $1.50 and specify UPS or first class mail. Illinois residents add 5% sales tax. Visa and MasterCard accepted.

Apple Game Paddles

John K. Elberfeld
Rochester, NY

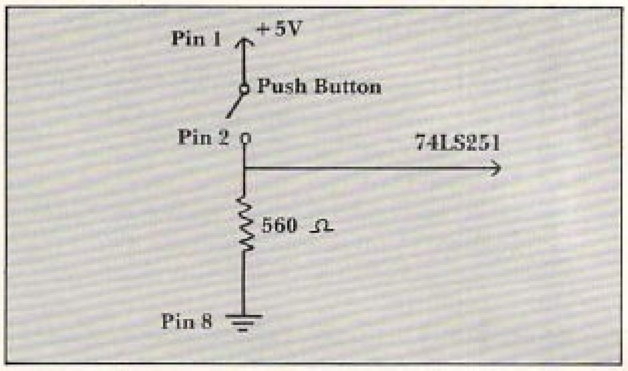
Understanding the functions of the Apple game paddles was my first step toward interfacing the Apple with the "real world." The simplicity of the paddles is a result of the amazing versatility of the game In/Out connector (where the game paddles are plugged in) on the computer mother board. The Apple Reference Manual, pages 23, 24, 100, and 114, provides the basic facts about the paddles and the connector. This article will attempt to unify these facts into an understandable explanation of the Integrated Circuit electronics used every time Little Brick Out is RUN.

The push button control on the paddles is a normally open momentary switch. No electricity may flow through this switch until it is pushed down. As soon as it is released, the flow of electricity stops. The source of voltage is pin 1 on the game I/O connector, which supplies +5 volts through the connecting wires to one side of the switch. When the button is pressed and the switch is closed, this +5 volts passes through the switch and back through the wires to pin 2 on the I/O connector. Pin 2 is connected to a standard Transistor-Transistor Logic (TTL) Integrated Circuit (IC) on the Apple mother board. When +5 volts is applied to pin 2, this IC causes the memory location of (-16287) or ($C061) to be greater than 128. The actual circuitry allows three push buttons to be used, but this article will supply details for button 0 only.

When pin 2 is grounded, the value of (-16287) will fall below 128. However, *not* applying +5 volts to pin 2 is *not* the same as grounding the pin. To make the pin grounded when no voltage is applied, a 560 ohm resistor is permanently connected between pin 2 and ground. Any electricity which might deceive pin 2 into thinking that 5 volts was being connected to it disappears through the resistor and into the ground. This resistor is located in the 16-pin DIP Header – the black box at the end of the paddle wires. This resistor is small enough to effectively ground pin 2 when no voltage is applied, but large enough so the current through it is very small when +5 volts is applied.

Figure 1 shows this information.

Figure 1. Push Button



The game controller analog inputs (knobs which turn) are physically very simple. The following details apply to PADDLE 0, but the concept is the same for all paddles. Just the pin numbers are different.

Pin 1 again supplies +5 volts through the connecting wires to one end of a 150,000 ohm variable resistor. The other end of the resistor is connected through the wires to pin 6. Turning the game control varies the resistance between the 5 volt supply voltage and pin 6 on the game I/O connector. When the resistor is turned so the resistance in the circuit is large, very little current can flow back to pin 6. A low resistance setting of the game control allows a large current to flow back to pin 6.

Pin 6 is a direct connection to a capacitor on the mother board. A capacitor is designed to store electric charge, but this capacitor starts out with *no* charge. Electric current which flows through pin 6 to the capacitor will build up the charge on the capacitor until it reaches a predetermined level. When the Apple is instructed to read the value of the game control through the BASIC command PDL(0), it uses a machine language monitor routine. This command starts a counting routine, consisting of a loop and a counter, to time how long it takes the electric current through the game controller resistor to charge the capacitor back up from its zero starting charge. The number of times that the computer executes this timing loop is the value reported when PRINT PDL(0) is RUN.

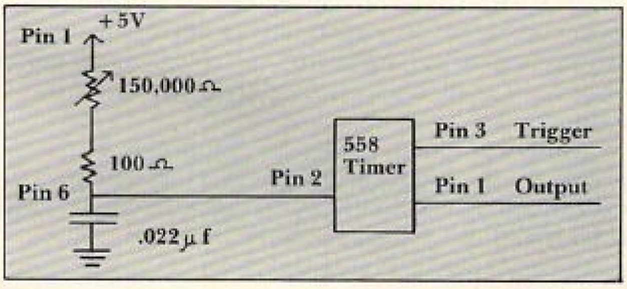
To limit the electric current flow into the capacitor when the game control is applying zero resistance, a small 100 ohm resistor is connected in series between the capacitor and the game controller. If too much current flowed into the capacitor in a very short time, the power supply or other circuits could be damaged.

How can the computer tell when a capacitor is charged? The Apple uses a 558 quad timer Integrated Circuit which is designed specifically to accomplish this. This circuit is similar to four 555 timers on one chip and allows up to four game paddles to be used at one time. The 558 IC is carefully designed to have a high output when the capacitor is charged to *less* than 2/3 of the supply voltage, and a low output after it reaches 2/3 of the supply voltage.

The timing cycle is started when the 558 is triggered – accomplished by PEEKing at location 49264 (-16272 or $C070). After the 558 is triggered, the output goes high, and the capacitor is allowed to charge up from the current through the game controller. When the capacitor reaches 2/3 of the supply voltage, the output goes low and remains low until triggered again. The 558 then automatically discharges the capacitor and waits for another trigger. Memory location $C064 is high (greater than 128) while the output of the 558 timer is high, and low (less than 128) when the output of the timer is low. How a high output from the 558 affects the value in a memory location is a good topic for some other author.

Figure 2 shows the circuit used for Paddle 0 on the Apple. The other game controllers have similar circuits but different memory locations.

Figure 2. Game Control Paddle



The Apple Reference Manual, Page 144, lists the following monitor routine for reading the value of the game controller. These directions are permanently stored in the Read Only Memory (ROM) of the Apple and are used during the execution of BASIC commands in regular programs. The following explanation assumes some knowledge of machine language programming.

```
PREAD  LDA  PTRIG
       LST  ON
       LDY  #$00
       NOP
       NOP
PREAD  LDA  PADDL0,X
       BPL  RTS2D
       INY
       BNE  PREAD2
       DEY
RTS2D  RTS
```

PTRIG is location $C070. LDA to this location triggers all 4 timers on the 558 chip on the mother board. This forces all the outputs of the 558 chip to go high and remain high until the capacitors are charged. This trigger also allows the capacitor to start charging.

LST ON is not a machine language command, but is probably a note to indicate that the timer has been triggered.

LDY #$00 loads the Y register with a value of zero to initialize it for the timing loop.

NOP means no operation. These are dummy commands used to take up some time before the timing loop is entered.

LDA PADDL0,X stores in the accumulator (A) the value of the memory location indicated by PADDL0, with the location address increased by the value stored in the X register. This allows a single routine to time all four allowable game controllers. The number of the controller (0 through 4) must be loaded in the X register before the routine is entered. PADDL0 is the memory location $C064, the memory location which is high when the current through paddle zero has not charged the capacitor. When register X has 0 stored in it, this location is read. When register X has the value "1" stored in it, the value in memory location $C064 + 1 = $C065 is loaded in the accumulator. $C065 is the memory location which is high when paddle 1 is in the process of charging its capacitor.

BPL RTS2D commands the computer to branch to location RTS2D if the value loaded into A is positive. RTS2D contains the command RTS which stops the timing loop and returns control to the BASIC commands which first called in the routine. A byte is considered negative when bit 7, the eighth bit from the right, is "1," and positive when this bit is "0." This seventh bit is also used as the flag to indicate that the output of the timer is still high. While the capacitor is in the process of being charged by the current through the game controller, this seventh bit in location $C064 is set to "1," the computer considers the value of the entire byte negative, and no branching occurs. The computer continues down the list of instructions. When the seventh bit is "0," it means the capacitor is charged enough, the timing stops with the value in Y indicating the number of times the program executed the loop, and the program branches to RTS2D which returns control back to the BASIC program. This is the normal END for this monitor subroutine.

INY commands the computer to increase the value of the Y register by 1. The Y register is now a counter which indicates how many times the computer passes this point in the loop. It is this value stored in Y which is the "value" of the game controller setting.

BNE PREAD2 commands the computer to branch back in the loop to the location of PREAD2 (the LDA PADDL0,X command) if the value of the Y register is *not* equal to zero. If the resistance is very high in game controller, the Y register may have counted up to 255, the largest number it can store. (All 8 bits are set to 1). Adding 1 more to 255 does *not* result in 256, but changes all the 1's to 0's (255 + 1 = 0). At this point the computer realizes it has gone too far, and it does not branch back to continue the loop.

DEY decreases the value of Y by one. Since Y must be zero to reach this point in the program, Y changes from 0 back to 255 because 000 - 1 = 255. This is the largest number that may be a paddle reading. At this point the timing loop is halted.

RTS then returns control to the BASIC program which called the subroutine. The value in Y is the value of the paddle reading.

The loop of:

```
READ2 LDA  PADDL0,X    (5 CYCLES)
      BPL  RTS2D       (2 CYCLES–NO BRANCH)
      INY              (2 CYCLES)
      BNE  PREAD2      (3 CYCLES–SUCCESSFUL
                        BRANCH)
```

is repeated until the capacitor is charged or until Y reaches a reading of 256 (000). The loop uses 12 clock cycles or about 12 microseconds. The maximum amount of time this program can measure is 255 loops or 3060 microseconds. The time needed to charge sufficiently the .022 mfd capacitor on the mother board through 150,000 ohm game controller is 1.1 * 150,000 * .022 E-06 = .0036 seconds = 3600 microseconds. This extra time guarantees that the maximum value can be reached when the game controller is set for maximum resistance.

Where can this information lead? It is possible to substitute a thermistor in place of the game controller so the reading of a paddle will indicate the temperature. A game controller with a resistance differing from the 150,000 ohm Apple Controller can be adapted to this circuit. A tilt sensitive switch could replace the pushbutton to indicate changes in position. The possibilities are endless and challenging. For more information on these circuits, see:

Mims, Forrest M.
Engineer's Notebook
Radio Shack, 1979

Lancaster, Don
TTL Cookbook
Howard W. Sams, 1974

Semiconductor Reference Guide
Radio Shack, 1981

©

In this interesting article, the author, a quadriplegic, describes techniques he has learned to effectively work with his computer. His suggestions may be of interest to other handicapped programmers and, at the end, he has a program-typing hint for everyone.

Computing Techniques For The Handicapped

George Leotti
Glenolden, PA

Being a physically disabled individual I faced some unique problems in making my computer, and peripherals, usable. First, I'd like to briefly describe my disability and then pass along some hints that may help other people, disabled and able alike.

I'm a C5-C6 quadriplegic. What that basically means is that my legs are totally paralyzed. My arms are partially paralyzed. I can't move my fingers or thumbs at all, but I can flex my wrists. Now, on to the hints.

There are many devices available to hold pencils, pens, and paintbrushes that will hold a stick which may be used to press keys on your keyboard. I use a universal holder with an L bar that holds a pencil, eraser-side down. Use whatever is comfortable for you.

To protect the faces on my small keyboard PET, I have someone cut out the fingers of a surgical glove, put them over the eraser, and tape it securely to the pencil. This way the eraser will not fragment and get stuck between the keys. Plus, wear and tear on the keys is reduced.

A New On-Off Switch

The first problem I encountered when I bought my PET was how to turn it on and off by myself. As you know, the PET and Commodore peripherals have their power switches located on the back panels. Why? I don't know. But I do know it is impossible for me to reach them.

The solution was simple. I had another switch mounted on the front panel, above the keyboard. The new switch "jumpers" the power switch in the rear. The Commodore switch now stays in the on position. The new switch turns power on and off.

A disaster-prevention technique for Apple Disk systems. Teachers should find this especially valuable. For DOS 3.3 with 48K RAM.

Apple DOS Changer

Robert Swirsky
Cedarhurst, NY

If there is an Apple in a classroom, there is usually a student who gets immense pleasure out of typing "INIT" and watching the drive whir and click as it erases the disk!

There is a strange phenomenon which occurs when you mix microcomputers and young people. Most of them will treat the computer with reasonable care, but there is always someone who feels compelled to try to "mess up" the computer's files.

I know from experience; I used to be such a student. What can be done to stop people from tampering with dangerous DOS commands? The answer is simple – just *change* the names of commands that you don't want available to users. Since the DOS is stored in RAM, it is relatively easy to find the table of commands, and modify it accordingly.

This program will change the DOS commands. When run, it displays each command followed by a prompt. If you want to change the command, just type in your revision. To leave it unmodified, hit the RETURN key. The next DOS command will be displayed, and the same procedure is followed.

A few words of caution are needed. First of all, don't make the new commands so that they match the beginnings of other commands. For example, if you change "CATALOG" to "C", conflicts will arise if you try to use the CLR, CLOSE, or other commands that begin with C. Second, don't put spaces in your command. This tends to make strange things happen. Also, avoid making your commands excessively long. There is only so much room in memory for the command table. Note, however, that the commands can be of different lengths from the original ones.

How To HAHA A Disk

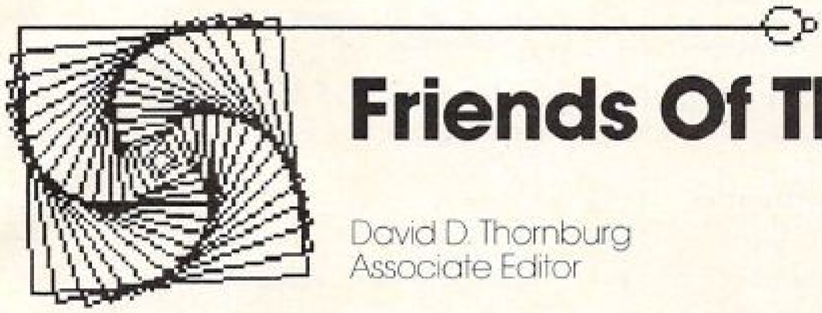
The program will work on DOS 3.3 with 48K of RAM. It will not work with less than 48K, without changing the addresses that the changes are POKEd to. Since the vast majority of Apple owners have the 48K, this is hardly a limitation.

After the program is RUN, the changes can be saved as a permanent part of DOS. Insert a *blank* disk, and type the INIT command, followed by the correct parameters. Of course, if you have changed the INIT *itself* to something else, you would type that in instead. When the initialization is completed, the disk will have your DOS modifications stored on it. Whenever DOS is booted from that disk, your custom commands will be the ones that the Apple knows.

A great deal of protection is offered with this program. If you decide to change INIT to HAHA, nobody can init the disk. Anyone who tries will be greeted with a ?SYNTAX ERROR (beep). However, when you go to HAHA your disk, you will be successful.

```
0 ONERR  GOTO 1000
1 DIM CO$(28)
2 P = 1
10 FOR X = 43140 TO 43271
20 CO$(P) = CO$(P) + CHR$ ( PEEK (X))
31 IF  PEEK (X) > 128 THEN P = P + 1
32 IF P = 28 THEN 70
40 NEXT
70 HOME
80 PRINT "DOS CHANGER BY ROBERT A. SWIRSKY
   "
160 VTAB 24: PRINT "HIT 'C' TO CONTINUE";:
    GET R$: IF R$ < > "C" THEN 160
170 HOME
180 PRINT : PRINT "YOU WILL SEE ALL 28 DOS
    COMMANDS"
190 PRINT "IF YOU WISH TO LEAVE IT UNCHANGE
    D,       PRESS RETURN"
200 PRINT "TO CHANGE THE COMMAND, SIMPLY RE
    TYPE IT WITH YOUR CHOICE."
210 PRINT : PRINT
220 LL = 43139
221 HL = 43271
230 FOR Q = 1 TO 28
235 PRINT CO$(Q);
240 PRINT  TAB( 20);" --> ";
250 INPUT R$
260 IF R$ = "" THEN R$ = CO$(Q)
261 IF  LEN (R$) = 1 THEN  PRINT "PLEASE --
     TWO OR MORE LETTERS": GOTO 240
270 FOR W = 1 TO LEN (R$) -1
280 POKE LL + W, ASC ( MID$ (R$,W,1))
290 NEXT
295 IFASC(RIGHT$(R$,1))>127THENPOKELL+W,ASC
    (RIGHT$(R$,1)):GOTO 501
300 POKE LL + W,ASC(RIGHT$(R$,1))+128
501 REM
510 LL=LL+W
520 IF LL > HL THEN 600
530 W=0
599 NEXT
600 PRINT "CHARACTER LIMIT EXCEEDED"
1000 PRINT "RUN TERMINATED"
```

A Monthly Column



Friends Of The Turtle

David D. Thornburg
Associate Editor

Dear Turtle ...

Dear Turtle,

I have an Apple II and want to buy LOGO for it. Which is the best version to get? My sister in Azuza (who has been chasing my husband since day one) likes the Terrapin version, and my husband (who would probably run off with the bank teller if I didn't have the goods on his father's secretary) favors Krell. I am leaning towards the LCSI version sold by Apple. Should I trust my heart?

Perplexed in Pensacola

Dear Perplexed,

As soon as I add Krell LOGO to my collection, I will write a report on all three. No matter which you choose, you will find LOGO to be a marvelous language. Your sister could be more of a problem if she knew how similar the Krell and Terrapin LOGO's are. I think it's time the three of you shared procedures

Dear Turtle,

Why is it that FOLLLK is now FOLLK (Friends of Lisp/LOGO and Kids) and is now located at 436 Arbalo Dr., San Francisco, CA 94132?

Just Asking

Dear Just Asking,

Beats me. It could be that they fellt that FOLLLK had tooooo many LLLLLLL's in it. Remember that a one-L lama is a priest, a two-L llama is an animal and that a three-L "lama" is a big fire.

Dear Turtle,

My brother-in-law runs a newsstand. Ever since he started selling **COMPUTE!** he has been too busy to see his wife (so he says). Since he is devoting his life to his work, what are the chances he will also be able to sell an all-LOGO magazine soon?

St. Louis Blue

Dear Saint,

My experts tell me that both the Krell people and the FOLLK folk will be publishing LOGO newsletters soon. If your brother-in-law also sold **COMPUTE! Books**, *he could build an addition to his newsstand to house your sister.*

Dear Turtle,

How come TI didn't go down the drain like Thornburg predicted when the 99/4 first came out?

Disillusioned

Dear Disillusioned,

TI's remarkable turnaround can be linked to the three L's: Low prices, Lots of product, and LOGO. TI is clearly in this business to stay. As for Thornburg ...

Dear Turtle,

Why haven't you told PET users about Spider by Bill Finzer? This "turtle" program is available from both the Softswap Microcomputer Center, San Mateo County Office of Education, 333 Main St., Redwood City, CA 94063, or from San Francisco State University, Center for Mathematical Literacy, 1600 Holloway Ave., San Francisco, CA 94132.

Tulare Fan

Dear Tulare Fan,

I just did.

Dear Turtle,

Talk about dumb. My local computer store salespeople don't know what a Big Trak is.

Amused

Dear Amused,

Tell them to drop into their local toy store to see the Big Trak robot vehicle by Milton Bradley. It is the best (and only) $40 programmable LOGO-like turtle on the market.

Dear Turtle,

I think Turtle Geometry is awful. It lets *anyone* quickly create pictures on a display with a minimum of training. People should have to *work* for this privilege. As for myself, I think people should use coordinate pairs and have to specify both the *x* and *y* coordinates for each point on the screen. Enough of this nonsense! I think, therefore I am most sincerely yours,

René Descartes

Dear René,

You might want to become a philosopher or something, and leave graphics to the rest of us.

Dear Turtle,

While you were answering those letters, I was trying out an Atari PILOT program written by Dr. R. Bharath, an associate professor at Northern Michigan University. His program simulates the "Drunkard's Walk" problem in which one guesses the number of steps needed to reach a target, given a certain probability of moving toward or away from the target. He hopes that **COMPUTE!** readers find it interesting.

Dr. T.

Dear Dr. T.

Dr. Bharath's program is presented here.

Confidential to Lost in Space,
As Atari PILOT programmer Richard Kline says,

```
GR: PEN RED

GR: 28(DRAW 6; TURN 13)

GR: PEN BLUE

GR: GOTO 12,-19

GR: TURN 14; 5(DRAW 5Ø; TURN 144)
```

```
1Ø T: DRUNKARDS WALK BY R. BHARATH
2Ø T:
3Ø T:A DRUNKARD COMES OUT OF PUB WHICH IS\

4Ø T:MIDWAY BETWEEN HOME AND A POND <TEN S
    TEPS FROM EACH>.  \
5Ø T:AT EACH STEP CHANCES ARE THE SAME THA
    T THE NEXT STEP WILL BE TOWARDS HO
    ME.\
6Ø T:YOU HAVE TO GIVE THIS PROBABILITY\
7Ø T:AND THE PROGRAM CALCULATES WHERE THE
    PERSON ENDS UP AND \
8Ø T:HOW MANY STEPS IT WILL TAKE TO REACH
    THE DESTINATION <HOME OR POND>.
9Ø T:
1ØØ T:
11Ø T:
12Ø T:WHAT ARE THE CHANCES THE NEXT STEP WI
    LL BE TOWARDS HOME?
13Ø T:<GIVE YOUR RESPONSE ON SCALE Ø TO 1Ø
14Ø T:1Ø MEANS CERTAIN TO TAKE EACH STEP TO
    WARDS HOME
15Ø T:Ø MEANS CERTAIN TO TAKE EACH STEP TOW
    ARDS POND >
16Ø A: #P
17Ø T:GUESS IF HE WILL REACH HOME<SAY YES O
    R NO>
18Ø A:$GUESS
19Ø T:GUESS HOW MANY STEPS HE WILL TAKE BEF
    ORE GETTING TO HIS DESTINATION <HO
    ME\
195 T:OR POND>
2ØØ A:#G
21Ø GR:TURN 9Ø
22Ø C:#D=Ø
23Ø C:#C=Ø
24Ø *BBB
25Ø GR:PEN YELLOW
26Ø GR:GOTO -5Ø,Ø
27Ø GR:GOTO 5Ø,Ø
28Ø GR: PEN RED
29Ø GR: GOTO #D,Ø
3ØØ C: #X=?\1Ø
31Ø C:#Y=1-<2*<#X<#P>>
32Ø T: STEP NUMBER #C
33Ø PA:5Ø
34Ø GR:GO 5*#Y
35Ø C:#D=#D+<5*#Y>
36Ø C:#C=#C+1
37Ø J<#D=-5Ø>: *EEE
38Ø J<#D=5Ø>: *EEE
39Ø PA :2Ø
4ØØ GR:CLEAR
41Ø J: *BBB
42Ø *EEE
43Ø PA:1ØØ
44Ø GR:QUIT
45Ø J<#D=5Ø>:*F
46Ø T:   REACHES HOME
47Ø A:=$GUESS
48Ø M:YES
49Ø TY:YOU WERE RIGHT
5ØØ TN:YOU GOT IT WRONG
51Ø J:*X
52Ø *F T:FALLS IN POND
53Ø A:=$GUESS
54Ø M:NO
55Ø TY:YOU WERE RIGHT
56Ø TN:YOU WERE WRONG
57Ø *X
58Ø T:TAKES #C STEPS IN PROCESS
59Ø T:<WITH CHANCE #P IN 1Ø OF TAKING A STE
    P TOWARDS HOME AT EACH STEP>
6ØØ T:
61Ø T:
62Ø T:
63Ø T:<YOUR GUESS WAS #G STEPS>
64Ø E:
```

©

For the Apple II, this simulates chemistry experiments for elementary students. It includes a list of choices and animated graphics.

Chemistry Lab

Joanne Davis
Kew Gardens, NY

"Chemistry Lab" encourages elementary school students to hypothesize and review concepts by allowing them to duplicate laboratory experiences in chemistry. It uses standard chemical indicators to identify a variety of substances as acids, bases, sugars, or starches.

The program is menu-driven. After choosing a topic, the student is shown instructions, followed by a picture of an eyedropper containing the indicator (in the appropriate color), a beaker (containing the material to be tested), and the material and indicator names. The student predicts the result of the test, as he or she would before conducting a laboratory experiment, and INPUTs the prediction.

When the test is carried out, the eyedropper releases its contents drop-by-drop and the beaker fills with liquid, its color indicating the presence of acid, sugar, etc. Comments then reinforce the material's classification.

This procedure is repeated to test four more substances. More items can easily be added by DIMensioning the arrays and adding more DATA.

Two Special Techniques

Two of the techniques used in this program should be of special interest. The animation is created by alternating between color and black and by time delays caused by empty FOR/NEXT loops. The inside of the dropper is blacked out a line at a time, with the delay making the action visible. The previous position of the drop is blacked out and then redrawn at a new location. Then the beaker is filled up, a line at a time.

Since the sugar and starch test required virtually the same instructions, a way had to be found to make the few alterations needed. The changes are READ in from DATA statements and inserted into the message.

A science curriculum can be made to come alive with animated laboratory experiments. Try it and see.

Table 1. VARIABLES

A$:	GET response
AN$:	Answer INPUT
B$:	Sugar or starch?
BEAK:	Color in beaker
C$:	Color name of indicator
CH$:	Menu choice
DROP:	Color in dropper
ID$, IH$, IR$:	Correct identity
IN$:	Name of indicator
N$:	Acid test materials
N1$:	Sugar/starch test materials
P:	Drop position
S,SO:	Sound
TT:	Time delay
X:	Counter
Y:	DATA flag

Table 2. SUBROUTINES

1000	Acid test
2000	Starch test variables
2500	Sugar test variables
3000	Sugar/starch instructions and tests
4000	READ data into array
5000	Graphics outline
6000, 6500	Animation

```
0 REM BY J. DAVIS
10 REM CHEM EXPER
20 TEXT : HOME
30 VTAB 8: HTAB 15: PRINT "***************"
40 HTAB 15:PRINT"WELCOME TO THE":HTAB 15:P
    RINT"CHEMISTRY LAB":HTAB 15:PRINT"
    ***************"
50 FOR TT = 1 TO 4500: NEXT
60 HOME : VTAB 5: PRINT "CHOOSE TEST 1, 2,
     OR 3:": PRINT : HTAB 5: PRINT "1.
     ACID":HTAB
65 HTAB 5: PRINT "4. QUIT"
70 GET CH$:CH = VAL (CH$): ON CH GOSUB 100
    0,2000,2500,100
80 TEXT : GOTO 60
100 END
1000 REM ACID/BASE***PHENOL
1020 REM INSTRUC
1025 GOSUB 4000
1030 TEXT : HOME
1040 PRINT:PRINT:PRINT "YOU ARE GOING TO TES
    T SOME MATERIALS TO": PRINT "SEE I
    F THEY ARE ACIDS
1041 OR BASES. THE": ? "INDICATOR WILL TURN 
    ";: INVERSE : ? "PINK";: NORMAL : ?
```

```
1042 " IN AN ";: INVERSE : PRINT "ACID";: NO
    RMAL : PRINT "."
1045 PRINT : PRINT "TYPE ";: INVERSE : PRINT
     "A";:NORMAL:PRINT"IF YOU THINK TH
    AT THE MATERIAL":
1046 PRINT "IS AN ACIS.": PRINT "TYPE ";: IN
    VERSE : PRINT "B";: NORMAL : PRINT
     "IF YOU THINK
1047 THAT THE MATERIAL": ? "IS A BASE."
1050 PRINT : PRINT "HIT ANY KEY TO BEGIN.":
    GET A$
1070 HOME
1150 FOR X = 1 TO 5
1152 DROP = 15: GOSUB 5000
1155 VTAB 21
1160 PRINT "INDICATOR: ";: INVERSE : PRINT "
    PHENOLPHTHALEIN"; NORMAL
1180 PRINT : INVERSE : PRINT "A";: NORMAL :
    PRINT "CID OR ";: INVERSE : PRINT
    "B";:
1181 NORMAL:PRINT "ASE ?":GET AN$: IF AN$ <
    > "A" AND AN$ < >"B" THEN HOME:GOT
    O1155
1190 IF ID$(X) = "A" THEN BEAK = 11: GOTO 12
    10
1200 BEAK = DROP
1210 GOSUB 6000
1220 HOME : IF ID$(X0 = "A" THEN PRINT N$(X)
    ;" IS AN ACID.": GOTO 1240
1230 PRINT N$(X);" IS A BASE."
1240 FOR TT = 1 TO 4000: NEXT TT
1250 NEXT X
1400 FOR TT = 1 TO 1000: NEXT TT
1500 RETURN
2000 Y = 1: GOSUB 4000: GOSUB 3000: RETURN
2500 Y = 2: GOSUB 4000: GOSUB 3000: RETURN
3000 REM STARCH/SUGAR INSTRUCTIONS
3010 TEXT : HOME
3020 PRINT : PRINT : PRINT "YOU ARE GOING TO
     TEST SOME MATERIALS TO": PRINT "S
    EE IF THEY
3021 CONTAIN ":B$(Y);".": ? : ? "THE INDICAT
    OR WILL TURN ";: INVERSE :?C$(Y);:

3022 NORMAL : PRINT " IN A ": INVERSE : PRIN
    T B$(Y);: NORMAL : PRINT "."
3030 PRINT:PRINT "TYPE ";: INVERSE :PRINT "Y
    ";: NORMAL:PRINT " IF YOU THINK TH
    AT THE MATERIAL":
3031 PRINT "CONTAINS ";B$(Y);" ."
3040 PRINT : PRINT "HIT ANY KEY TO BEGIN.":
    GET A$
3050 HOME
3060 FOR X = 1 TO 5
3070 DROP = P(Y): GOSUB 5000
3080 VTAB 21
3090 PRINT "INDICATOR: ";: INVERSE : PRINT I
    N$(Y): NORMAL
3092 :
3094 :
3096 :
3100 PRINT "NOW TESTING: ";: INVERSE : PRINT
     N1$(X): NORMAL
3110 PRINT : INVERSE : PRINT B$(Y);: NORMAL
    : PRINT " (Y/N) ?": GET AN$
3120 IF AN$ < > "Y" AND AN$ < > "N" THEN HOM
    E : GOTO 3080
3130 ON Y GOSUB 3500,3600
3140 FOR TT = 1 TO 4000: NEXT TT
3150 NEXT X
3200 RETURN
3500 REM STARCH MESSAGE
3510 IF IH$(X) = "S" THEN BEAK = 3: GOTO 353
    0
3520 BEAK = DROP
3530 GOSUB 6000
3535 HOME
3540 IF IH$(X) = "S" THEN PRINT N1$(X);" CON
    TAINS STARCH.": GOTO 3560
3550 PRINT N1$(X);" DOES NOT CONTAIN STARCH.
    "
3560 RETURN
3600 REM SUGAR MESSAGE
3610 IF IR$(X) = "S" THEN BEAK = 9: GOTO 362
    5

3620 BEAK = DROP
3625 GOSUB 6000
3627 HOME
3630 IF IR$(X) = "S" THEN PRINT N1$(X);" CON
    TAINS SUGAR.": GOTO 3660
3650 PRINT N1S(X);" DOES NOT CONTAIN SUGAR."

3660 RETURN
4000 RESTORE
4005 FOR X = 1 TO 5
4010 READ N$(X),ID$(X),N1$(X),IH$(X),IR$(X)
4020 NEXT X
4030 FOR X = 1 TO 2
4040 READ B$(X),C$(X),IN$(X),P(X)
4050 NEXT X
4060 RETURN
5000 REM SCREEN**OUTLINE BEAK AND DROP
5010 GR : COLOR= 10
5020 VLIN 0,20 AT 14: VLIN 0,20 AT 18
5030 HLIN 15,17 AT 0: HLIN 13,19 AT 6
5040 HLIN 15,17 AT 21: VLIN 21,24 AT 16
5050 PLOT 9,28: VLIN 28,38 AT 10
5060 VLIN 28,38 AT 21: HLIN 11,20 AT 38
5065 REM INSIDE DROPPER
5070 COLOR= DROP
5080 VLIN 15,20 AT 15: VLIN 15,20 AT 16: VLI
    N 15,20 AT 17
5500 RETURN
6000 REM ANIMATION
6010 COLOR= 0
6015 P = 31:S = - 16336
6020 FOR G = 15 TO 20
6025 PLOT 16,P
6030 HLIN 15,17 AT G:SO = PEEK <S> - PEEK <S
    > - PEEK <S>
6033 GOSUB 6500
6035 FOR TT = 1 TO 400; NEXT TT
6037 COLOR= 0
6040 NEXT G
6050 FOR TT = 1 TO 400; NEXT TT
6100 COLOR= BEAK
6110 FOR G = 37 TO 32 STEP - 1
6120 HLIN 11,20 AT G: FOR TT = 1 TO 250; NEX
    T
6130 NEXT G
6140 RETURN
6500 COLOR= DROP:P =P + 1: PLOT 16,P
6510 RETURN
7000 DATA SOAP,B,BREAD,S,S,LEMON JUICE,A,CRA
    CKER,S,0,COLA,A,CHOCOLATE,0,S,BAKI
    NG
7001 SODA,B,COLA,0,S,VINEGAR,A,FLOUR,S,0
7010 DATA STARCH,PURPLE,IODINE,13,SUGAR,ORAN
    GE,BENEDICTS SOLUTION,7
30000 REM BY J. DAVIS
```

©

Apple Manager: An Alphanumeric Data Manager

Robert Jacques Beck
Minneapolis, MN

I began writing a data management program as part of a classroom assignment, but I finished it only because I had become obsessed with fitting the pieces of the puzzle together. I learned that the ideal data management system does everything under the sun and will never be invented. The data manager described in this article is designed primarily for string data, although numeric applications are possible.

It has two advantages. First, you get a listing. Second, it's written in BASIC and you get some explanation of how it works. If you find that it doesn't meet all your needs, and you don't want to modify it, you'll have some valuable knowledge if you go looking for another data manager. The program is written for the Apple – hence the name Apple Manager – but many ideas collected in it can be used elsewhere. The rest of this section is a somewhat theoretical discussion, so if you want to get into the particulars of the program just skip ahead.

Computers are great at keeping track of large masses of information. That's what data management is all about, so asking "Why do we need data managers?" is like asking "Why do we need computers?" But that is a kind of circular definition. Maybe we should ask, "What should a data manager program do?"

I like to think of data managers as two-way transportation systems between my diskettes and me or, more technically, between the storage device and the information source. To store data we must input it, but that's not enough to give us control over the contents of our data files: we might want to come back later and modify or delete something. Similarly, information retrieval is not just a matter of pulling the stuff out as fast as we can; we may be interested in one kind of information on Monday and another on Tuesday. You'll see flexibility come alive when you try the program.

From Aardvarks to Ziggurats

Since data is stored in files, a data manager is first and foremost a file manager. You can use the same data manager program to deal with files from Aardvarks to Ziggurats because each file will have the same general structure, even though individual components may vary. You may not think about it, but you will definitely take advantage of similarities in file structure when you write additions to a data manager or interface your files with other programs.

Just as books are made up of pages, files are made up of records. Records can be subdivided into fields, much like the sentences on a page. So we can define a record as a logical grouping of several individual data items. If each record in a file is identical (that is, if it has the same size and makeup), several records hypothetically placed adjacent to each other will look like a rectangle. A rectangular file structure is easy to program.

Another possibility is a hierarchical structure. Hierarchical files have records that are built from the same group – but not necessarily the same number – of components. Hierarchies occur naturally in many applications. Suppose you want to keep tabs on the books in your library and you want to cross-index them by one or more topics. One approach is a rectangular file with each record storing information about one book.

But there's a slight problem. You will need a field for author, another for title, and one additional field for each topic. Because you need to know how many fields to allot, you'll have to decide in advance on a reasonable maximum number of topics. On the other hand, a hierarchical file doesn't lock you into a fixed design. Imagine a hierarchy with author at the top. Titles are second in status, with each title being linked to its author's name. Topics are linked, in turn, to titles. In both cases, there is no set number of linkages.

Though this program is based on a rectangular organization, I just wanted to point out that there are alternative ways to set up data bases.

Module Structure

The more a program does, the more likely it is to grow to an unmanageable size. One way to cope with this is to break your program into chunks called modules. The main driver (lines 91-92) prints a menu and lets your choose one of the five modules: Files, Records, Reports, Select File, or Utilities. With the exception of Select File, all of the modules are multi-functional so the first thing you see when you enter them is another menu.

Menus allow you to move away from the main drive and into the tangled depths of the program, but how do you return? Whenever Apple manager requests input, if you type CONTROL O (for Out), plus a RETURN if necessary, you'll back up one level in the program hierarchy. This is a handy escape from any operation. It's the only way you exit from functions that don't automatically return to menu. To switch from one module to another, type CONTROL O to back up to the main menu, then select the new module.

To conserve memory, I used a lot of subroutines and multiple statement lines. Program lines are numbered consecutively, instead of adding 10 to each line number as is usually done. There are so many GOSUBs and GOTOs that I saved about 300 bytes this way. Variable names are short and variables are used and reused whenever possible. Look at Table 1 to sort out some of the confusion.

Diskette Data Bases

Apple Manager assumes every diskette is an independent filing system. Each diskette has one title file, which is a sequential file containing the number of data files on the diskette and their names. When the program starts up it reads the names into an array (T$, line 89). Whenever a file name is input later on, it can be checked against this array to see if it is a valid file name for the diskette (lines 57-60). Data is kept in random access files. The data manager has to know how to relate to these files. Somebody has to tell it things like what size record to use, how many fields, what their names are, and so on. The easiest way to do it is to put the information into a file.

Rather than put it all in one file, I set up a separate file for the description of each data file. The description file is read into two arrays, one containing the names of the fields and the other containing the field lengths in characters (or bytes, since one character is stored per byte). You might want to look at the Atari Data Manager in **COMPUTE!** (November, 1981, #18) where the same concept is implemented somewhat differently. Fields are referred to as "items" by Apple Manager, so I'll use the two terms interchangeably. To summarize, each diskette has one title file, and for each data file there will be a description file.

Files

Most of this module is pretty easy to use once you get it running. The Catalog function is simple: it lists the title file array (lines 99-100). Describe File is similar in that it prints the description file arrays (lines 101-103). Create File is a bit more complicated – here's where new file structures are born. First type in the file name, then the number of items per record. All items are alphanumeric, in other words: strings. (Numbers are stored on diskette as strings anyway, one byte per number, because that's how Apple DOS formats diskette storage.)

After you finish entering a name and length for each item, you'll fall into the file editor. Record length is the sum of the field lengths plus the number of fields (because there is a return character after each field).

The file editor (lines 113-131) is basically a list editor. In BASIC, lists are virtually synonymous with arrays. It is the arrays holding the file description (L$ and L%) that get manipulated here. The edit menu uses abbreviations. (Replace the pound sign [#] with an integer.) I suggest you make a few simple typographical errors to see how the program responds. This is what the abbreviations mean:

S – saves/creates a data file and a description file.

R – review. Prints the file description.

A# – add # new items to the description (i.e., A3 = add three items).

D# – delete item # (i.e., D2 = delete item number 2).

I# – insert an item into position #.

N# – change the name of item number #.

L# – change the length of item number #.

Deleting and inserting items is done by shifting both description arrays; changing a name or length is done by entering a replacement for an element of one of the two arrays.

I once read somewhere that file maintenance consists of content changes and structural changes. The record editor described below takes care of content changes. *Evolvability*, the capacity to respond to changing needs, is accomplished through the file editor. If a check (line 114: is B>0?) shows that the file has data in it, you can still edit the file structure, although the program works a little harder. First, the original description is copied into some temporary arrays (line 114 again). When you're done monkeying around and you choose the Save option, the old and new descriptions are compared (lines 129-131). Next a scratch, or temporary, file is written to meet the new specifications (line 131).

Adding new items or changing an item name presents no problem. If an item is deleted it won't be rewritten; if an item is shortened, any instance of it that's too long gets truncated from the right. After the scratch file has been successfully completed, the old file is deleted and the scratch file is renamed. Apple Manager uses scratch files in a couple of other places, namely when sorting a file or deleting records. These routines also write a new, updated file before deleting the old file. You could run into a DISK FULL error if there weren't enough room. By the way, I've chosen the unlikely name of "A control D" for the scratch file's name, so it shouldn't interfere with any of your files.

The Other Choices

Perhaps you've asked yourself, "How does Apple Manager know when a file has data?" The method is simple. Record zero, the first record in a random access file, stores the number of records. This number is updated whenever records are added or deleted (line 64). A newly created or emptied file (see below) is actually one record on the diskette with a 48 (the ASCII code for zero) in the first byte.

No data manager would be complete without the ability to get rid of unwanted files. Apple Manager deletes the data file and the corresponding description file and removes the file name from the title array (lines 132-134). The title array may change several times in one run. Rather than rewrite the file each time, a flag variable, F, is set. The title file on the diskette is updated (line 66) when you exit the Files module – if the flag is set (see lines 5 and 223). This is why the disk drive light may come on when you switch modules.

If you don't want to remove a file (just reuse it), then the Empty function at lines 135-136 is for you. This section deletes a data file but not its description file, and opens a new data file of the same name. The net effect is to empty a file of data while preserving the file structure. Copy creates a new data file by using an already existing description file.

A few words about limits and error handling are undoubtedly in order here. The dimension statement in line two defines the first three of these somewhat arbitrary limits, so they can be easily changed:

25	files per diskette.
50	fields per record.
1000	records per file.
115	bytes = maximum field length.
20	characters maximum in a file or item name.

The last two limits, as well as many errors, are avoided by checking input: line 32 (Is a number out of range?), line 35 (Has the return key been pressed without first typing something?) and line 36 (Is a string too long?), are examples. But what if you try to make the program do something illegal, such as read data that doesn't exist? ONERR is meant for just such cases, though it does have the drawback of stopping the Apple's excellent error messages.

Here's how I compromised. If there's an error in line 89 – e.g., if there's no title file because it's the first time you're using a diskette – you jump to line 90 where the error flag for the rest of the program is set. From here on in, unless your error is one of the DOS errors dealt with by lines 220 and 221, the POKE 216,0 at line 222 cancels ONERR. RESUME causes the error to recur so you get an Apple message. Control C (program interrupt; error code =255) is handled differently; it still works, but without the expected BREAK message.

Using A Directory

Let's assume we've got an imaginary file defined and that data has been entered into it. Let's also assume we want to extract information about an author named Kilroy. One way to do it is to search the entire file until we find the Kilroy record. But disk access is slow and we are impatient, so let's use a directory to locate the record instead.

When a file is selected, Apple Manager opens the description file and reads it into arrays (line 67, called as a subroutine from line 94). The data file is also opened and the first field of every record is read into the array D$ (line 94). It's faster to search this directory array than to search the file. There is a one-to-one correspondence between array elements and records (Figure 6); if *Kilroy* is the seventh array element then record seven is the record we want.

What we have done is to define the first field – in this case *author* – as the record identifier. Record IDs are used for rapid access in Delete, Print, and Change (described below). A subroutine beginning at line 43 requests the ID and searches the directory. You don't have to enter the complete ID. For instance, repeatedly typing "K" will locate, in sequence, all authors whose name begins with K.

Assuming 48K of memory, there are about 17,000 free bytes for the directory. (The exact amount depends on how much is used up by the title array, the description array, and other string variables.) If the first item is a long one, it may not be possible to have 1000 records in the file without disabling the directory. The same memory problem may arise when you sort, since the D$ array holds the item being sorted.

You now have an outline of how the program works. We could step slowly and leisurely through the code, but I don't want to send the editors into apoplexy. The rest of this writeup is a guide to using the program. A good way to start is by creating a file or two. Then go to the Records module, enter some fictitious data, sort it, and edit it. Next try a report. Then go back to Files and change the file structure. Now generate another report to see how stored data has been affected.

Apple Manager makes a good, if rudimentary, stab at most data management functions. One omission is computed variables. This is not a short program, so I'll make copies for anyone who sends $3, a diskette, and a stamped, self-addressed mailer to: Robert Beck, 2101 21 Ave. S., #W15, MPLS, MN 55404. To those who are typing it in, I wish a steady hand and a steadier eye. Happy data managing.

Records Module

ENTER (lines 147-150) – Initializes each item to an asterisk (*) – so missing data is not a problem – then goes to the record editor (lines 8-27), which has five options:

1) Retype – type in a new value.
2) Control O – exit to menu.
3) Control B – back up one field in the file.
4) Control F – forward to next record.
5) RETURN – the return key must be pressed after each of the above options. Pressing the return key alone does not affect the item displayed; it moves you to the next item in the record.

Record number appears in parentheses to the left of the item name. Use Control B to backspace through a file; use RETURN to move forward through a file.

DELETE (lines 139-146) – marks records you choose to delete by placing a Control E in the first byte. When you exit this option via Control O, Apple Manager rewrites the file without marked records.

CHANGE (lines 151-158) – allows a "window" in each record to be set by selecting a starting and an ending item number. Once a starting record in the file is chosen, the record editor, which works as previously described, is called.

PRINT (lines 159-160) – prints one record at a time. This is the fastest way to retrieve a record (Figure 7).

SORT (lines 161-165) – reads the sort key (item to sort by) into the array D$, then sorts, in either ascending or descending order, by the bubble method. It's really a tag sort because the record numbers (in S%) are also sorted. Once record numbers are properly ordered, the file is rewritten. The sort is alphanumeric, so "17" is placed before "7" and after "07".

Report Module

1. Retrieve (lines 169-171, 190-198) – formats the report in tabular form if the printout from one record fits on one line (Figure 8), otherwise prints one item per line. A variable number of fields can be retrieved, and in any order.

2. With Sums – same report as Retrieve, with the addition (pun intended) that a sum for each item is printed.

3. Frequencies (lines 199-204) – counts the number of times each value of an item occurs. First, sort the file by that item.

4. Case Selection (lines 172-188) – a technique that lets you retrieve information by its characteristics – you can pick out a subset of the file. All you have to do is input selection criteria in the form of minimum and maximum item values. The values are stored in arrays (M$ and N$) and may be ORed together in groups of five or less. Up to five of these groups can be ANDed together:

(a1 OR b1 OR c1 OR d1 OR e1) AND ... AND (a5 OR b5 OR c5 OR d5 OR e5).

Each a1, b1, c1, etc. is of the form MIN(A)<=A=<MAX(A).

Got that? Well, here's how to work it:

1) Select an item by typing its number.

2) Select a range for that item by typing a minimum and a maximum. Pressing the return key without typing anything sets a minimum to the null string or a maximum to CHR$(95).

3) Terminate a series of ORs and go on to the next AND by typing a slash (/).

4) Terminate the whole thing at any point, including the very beginning, by typing a period.

Note: Each record is checked against the criteria stored in arrays (lines 76-79) and ignored if it doesn't meet them (i.e., if G=5). For an alternative, and very interesting, method of introducing changeable functions into your program, see "Algebra String – a Self-altering Program" in **COMPUTE!** (September, 1981, #16).

Utilities Module (lines 212-218)

Upload – Each record's fields are laid down sequentially within the record. There will be null bytes at the end of the record if any item is shorter than its defined byte length. Upload removes all null bytes from a file by adding blanks where needed. The name "Upload" comes from the fact that some mainframe computers interpret null bytes as end-of-record marks; to send diskette files to them null bytes must be removed.

Download – Does the opposite of Upload.

Drive Select – Use to switch from one disk drive to another. Each diskette remains an independent data base.

```
1  REM    APPLE MANAGER   **  ROBERT JAC
     QUES BECK  **
2 D$ =  CHR$ (4): FOR I = 768 TO 777: RE
     AD J: POKE I,J: NEXT : DIM L$(50),
     F$(4),T$(25),L%(50),D$(1000),R$(50)
     ,S%(1000),S(50),M$(25),N$(25): FOR
     I = 0 TO 4: READ F$(I): NEXT : GOTO
     88
3  VTAB 23: PRINT "PRESS RETURN TO CONT
     INUE";: CALL  - 958
4  POKE  - 16368,0: INPUT "->";Z$:H =
     RIGHT$ (Z$,1) =  CHR$ (15)
5  IF E = 7 AND F = 1 AND H THEN  GOSUB
     66
6  IF H THEN  POP : CALL 768: ON E GOTO
```

```
     96,104,166,212,137,152,91,115,144,
     156,172
7 Z = VAL (Z$): RETURN
8  FOR Y = O TO K
9  VTAB 19: HTAB 1: PRINT " (R"I")" SPC
     ( 2)Y". "L$(Y)":": PRINT : PRINT R
     $(Y)
10  HTAB 1: VTAB 21: INPUT "";Z$: IF Z$
```

Table : Partial list of variables.

Name	Description
A$	File name.
D$	Control D (DOS commands).
D$(1000)	Directory and sort array.
F$(4)	Option titles (Records).
L$(50)	Item (field) names.
L%(50)	Item lengths.
M$(25)	Minima of criteria (Report).
N$(25)	Maxima of criteria (Report).
R$	Current value of item being counted (frequencies).
R$(50)	Fields of current record.
S(50)	Item sums (Report).
S%(1000)	Sort array (holds record numbers).
S%(1-50)	Item numbers being reported.
S%(200-250)	Criteria item numbers (Report).
S%(400-450)	Spaces allotted to an item in report printout.
Z$	Input.
A	Number of fields per record.
B	Number of records.
C	Maximum permissible value.
	Frequency count of an item (frequencies).
D	Input error flag.
	Number of lines on screen.
E	Module flag.
F	Title file update flag.
G	Option flag (Files).
	G<>5 if record meets criteria (Report).
	Total sum (Report).
H	Control 0 flag.
I,J,K,Q	Temporary indices.
K	Ending item number (edit window).
L	String or item length.
N	Number of files.
O	Starting item number (edit window).
	Number of criteria (Report).
P	P = 1 if file name is in title file.
	Number of dashes to print.
Q	Record number.
	Larger of item length and length of item name (Line 74).
R	Record length.
S	Option selected (Report).
V	Drive number.
	Number of printer columns (Report).
W	Slot number.
X	Number of spaces needed for tabular report.
	First record to edit (Records).
Y	Number of items being reported.
	Item number being counted (frequencies).
	Error code.
Z	Numeric value of input.

```
     = "" THEN 15
11  IF Z$ =  CHR$ (2) THEN 23
12  IF Z$ =  CHR$ (6) OR Z$ =  CHR$ (15
     ) THEN X =  ASC (Z$): GOTO 18
13 L = L%(Y): VTAB 17: GOSUB 36: IF D T
     HEN 10
14 R$(Y) = Z$: VTAB 18: CALL  - 868
15  GOSUB 27: NEXT Y
16 Q = I: GOSUB 48:D$(I) = R$(1): IF X
      = 15 THEN  GOSUB 64: CALL 768: GOTO
      137
17  RETURN
18  IF V = 6 THEN 16
19 H = Y = 1 AND I = B + 1 AND R$(1) =
      "*": IF H AND X = 6 THEN 10
20  IF H AND X = 15 THEN  GOSUB 64: CALL
      768: GOTO 137
21  IF X = 15 THEN B = B + 1
22  GOSUB 27: GOTO 16
23  IF I = 1 AND Y < 2 THEN  VTAB 21: PR
     INT R$(Y): GOTO 10
24  IF Y > 1 THEN  GOSUB 27:Y = Y - 1:
     GOTO 9
25 Q = I: GOSUB 48:D$(I) = R$(1):I = I -
   - 1: IF V = 5 THEN  GOSUB 27:Y = K:
     GOSUB 47: GOTO 9
26  VTAB 6: CALL  - 958: PRINT "* ID= "D
     $(I): POKE 34, PEEK (37) + 1:Y = K:
      GOSUB 47: GOTO 9
27  HTAB 1: VTAB 21: PRINT R$(Y): FOR E
      = 1 TO  LEN (R$(Y)) / 40 + 4: CALL
      - 912: NEXT : RETURN
28  PRINT "** WHICH?";: POKE  - 16368,0:
       GET Z$: IF Z$ < >  CHR$ (15) THEN
       Z =  VAL (Z$): RETURN
29 H = 1: GOTO 5
30 C =A
31  GOSUB 4
32  IF Z < 1 OR Z > C OR  INT (Z) < > Z
       OR ( VAL ( RIGHT$ (Z$,1)) =0 AND
       RIGHT$ (Z$,1) < > "0") THEN PRINT
        "TYPE AN INTEGER FROM 1 TO "C;:W
        = 1: GOTO 31
33  RETURN
34  GOSUB 4
35 D = 0: IF Z$ = "" THEN  PRINT : PRINT
       "TYPE SOMETHING BEFORE PRESSING
       RETURN!":D = 1: RETURN
36 D = 0: IF  LEN (Z$) > L THEN  PRINT :
       PRINT "THAT IS TOO LONG, TRY AG
       AIN":D =1
37  RETURN
38 I =  VAL ( MID$ (Z$,2)): IF (I > A)
       THEN PRINT "NUMBER OUT OF RANGE-
        START OVER!": POP : GOTO 116
39 I =  VAL ( MID$ (Z$,2)): IF I = 0 OR
       ( VAL ( RIGHT$ (Z$,1)) = 0 AND R
       IGHT$ (Z$,1) < > "0") THEN PRINT
    .   "TYPO, TRY AGAIN!": POP : GOTO
       116
40  RETURN
41  PRINT : PRINT "NAME OF ITEM "I"?";:
       GOSUB 34:L$(I) = Z$: IF D = 1 TH
       EN 41
42  PRINT "ITEM LENGTH?";:C = 115: GOSUB
       31:L%(I) = Z:R = + Z + 1: RETURN
43  PRINT :L = 115: PRINT "RECORD ID?";:
     GOSUB 34:U = I:J = LEN (Z$): IF
     D = 1 THEN 43
44  IF U = B THEN U = 0
45  FOR I = U + 1 TO B: IF  LEFT$ (D$(I)
     ,J) < > Z$ THEN  NEXT : IF U > 0
     THEN U = 0: GOTO 45
46  IF I > B THEN  PRINT : PRINT "ID NOT
     FOUND, TRY ANOTHER!": GOTO 43
47  PRINT D$"READ"A$",R"I: FOR J = 1 TO
     A: INPUT R$(J): NEXT : PRINT D$:
     RETURN
48  PRINT D$"WRITE"A$",R"Q: FOR J = 1 TO
     A: PRINT R$(J): NEXT : PRINT D$:
     RETURN
49  PRINT D$"WRITEA"D$",R"Q: FOR J = 1 T
     O  A: PRINT R$(J): NEXT : PRINT
     D$: RETURN
50  IF W = 0 AND D / 15 < >  INT (D / 15
     ) THEN  PRINT : PRINT : GOSUB 3:
     VTAB 23: HTAB 1: CALL  - 868
51  RETURN
52  PRINT
53 D = D +1: IF W = 0 AND D / 15 =  INT
     (D / 15) THEN  PRINT : PRINT : P
     RINT : GOSUB 3: VTAB 21: HTAB 1:
     CALL  - 958
54  RETURN
55  PRINT L$(Q)": "R$(Q): FOR J = 1 TO
     INT (( LEN (L$D(Q) + R$(Q))) / 4
     0) + 1: GOSUB 53: NEXT :RETURN
56 M$ = D$(J):D$(J) = D$(J - 1):D$(J - 1
     ) = M$:K = S%(J):S%(J) = S%(J -
     1):S%(J - 1) = K:K = 1: RETURN
57  PRINT : PRINT "FILE NAME?";:L = 20:
     GOSUB 34: IF D = 1 THEN 57
58 A$ = Z$: FOR I =1 TO N: IF Z$ = T$(I)
     THEN P = 1: RETURN
59  NEXT :P = 0: IF G < 2 THEN  PRINT :
     PRINT Z$" ISN'T IN THE TITLE": P
     RINT "FILE, TRY AGAIN.": GOSUB 3
60  RETURN
61  GOSUB 63: IF B >  0 THEN  PRINT : P
     RINT "FILE "Z$" HAS DATA -": IF
     P < 3 THEN  POP : GOSUB 3: GOTO
     96
62  RETURN
63  GOSUB 65: PRINT D$"READ"A$",R0": IN
     PUT B : PRINT D$: RETURN
64  GOSUB 65: PRINT D$"WRITE"A$",R0": P
     RINT B: PRINT D$: RETURN
65  PRINT : PRINT D$"OPEN"A$",L"R: RETU
     RN
66  PRINT : PRINT D$"OPEN TITLE FILE":
     PRINT D$"WRITE TITLE FILE": PRI
     NT N: FOR I = 1 TO N: PRINT T$(
     I): NEXT : PRINT D$"CLOSETITLE
     FILE":F =0: RETURN
67  PRINT D$"OPENDES "A$: PRINT D$"READ
     DES "A$: INPUT A,R: FOR I = 1 T
     O A: INPUT L$(I),L%(I): NEXT :
     PRINT D$"CLOSE": RETURN
68  HOME : PRINT "FILE NAME: "A$: PRINT
     : PRINT A" ITEMS PER RECORD": P
     RINT : PRINT "RECORD LENGTH: "R
69  PRINT : PRINT "#" SPC( 6)"ITEM" SPC
     ( 20)"LENGTH":P = 40: GOSUB 82:
     PRINT : RETURN
70  GOSUB 68: FOR I = 1 TOA: PRINT I"."
     SPC( 6 -  LEN ( STR$ (I)))L$(I)
     SPC( 26 -  LEN (L$(I)))L%(I): P
     RINT : IF (I - 6) / 8 =  INT ((
     I - 6) / 8) AND (I < > A) AND W
     = 0 THEN  GOSUB 3: HOME : GOSUB
     69
71  NEXT : PRINT D$"PR#0": RETURN
72  FOR W = U TO O * 5:M$(W) =  CHR$$ (
     95):S%(200 + W) = 0: NEXT : RET
     URN
73  VTAB 23 - C: CALL  - 958: PRINT : R
     ETURN
74 J = S%(I):Q = L%(J): IF  LEN (L$(J))
     > L%(J) THEN Q =  LEN (L$(J))
75 X = Q + X + 2:S%(400 + I) = Q: RETURN
76 G = 0: GOSUB 47: IF O = 0 THEN  RETU
     RN
77  FOR K = 1 TO O:G = 0: FOR J = 1 TO
     5:U = (K - 1) * 5 + J:Q = S%(20
     0 + U): IF R$(Q) < M$(U) OR R$(
     Q) > N$(U) THEN G = G + 1
78  NEXT : IF G = 5 THEN  RETURN
79  NEXT : RETURN
80  PRINT : PRINT "SEND PRINTOUT TO SLO
     T NUMBER?":P = 12: GOSUB 82: PR
     INT "DEFAULT = TV": GOSUB 82: V
     TAB  PEEK (37) - 3: HTAB 30 : G
     OSUB 4:W = Z: CALL  - 958: RETU
     RN
81  PRINT D$"PR#"W: RETURN
82  FOR J = 1 TO P: PRINT "-";: NEXT :
     PRINT : RETURN
83  HOME : VTAB 2: PRINT  TAB( 13)F$(Z
     - 1)" RECORDS"
84  VTAB 4: PRINT "FILE NAME: "A$: POKE
     34,5: VTAB 7: RETURN
85  IF B = 0 THEN  RETURN
86  FOR J = 1 TO V: IF L$(I) < > D$(J)
     THEN  NEXT : RETURN
87 D$(J) = Z$: RETURN
88 V = 1: ONERR  GOTO 90
89  PRINT D$"OPENTITLE FILE,D"V: PRINT
     D$"READTITLE FILE": INPUT N: IF
     N > 0 THEN  FOR I = 1 TO N: INP
     UT T$(I): NEXT
90  PRINT D$"CLOSE": ONERR  GOTO 219
91  TEXT : HOME : VTAB 2: PRINT  SPC( 1
     0)   APPLE MANAGER   ": VTAB 7:
     HTAB 3: PRINT "1  FILES" SPC( 1
     2)"4  REPORTS": PRINT : HTAB 3:
     PRINT "2  SELECT FILE" SPC( 6)"
     5 UTILITIES": PRINT : HTAB 3: P
     RINT "3  RECORDS" SPC( 10)"6 QU
     IT"
92 E = 7: VTAB 20: GOSUB 28: ON Z GOTO
     96,93,137,166,212,223: PRINT :
     PRINT "YOU CAN'T CHOOSE THAT! T
     RY AGAIN": GOTO 92
93 O = 1:E = 7: HOME : VTAB 4:G = 0: PR
     INT  TAB( 11)"-- SELECT A FILE
     --": VTAB 6: GOSUB 57: IF P = 0
     THEN 93
94  GOSUB 67: GOSUB 63: IF B > 0 THEN F
     OR I = 1 TO B: PRINT D$"READ"A$
     ",R"I: INPUT D$(I): NEXT : PRIN
```

```
       T D$
95  GOTO 91
96 O = 2:L = 20:G = 0:P = 0: HOME : VTA
       B 2:E = 7: PRINT  SPC( 13)" <<
       FILES  >>": VTAB 6: PRINT : PRI
       NT "1  CATALOG" SPC( 10)"5 EDIT
       DESCRIPTION": PRINT : PRINT "2
       DESCRIBE FILE" SPC( 4)"6  EMPTY
       FILE": PRINT : PRINT "3  CREATE
       FILE" SPC( 6)"7  COPY DESCRIPTI
       ON"
97  PRINT : PRINT "4  DELETE FILE" SPC(
       6)"8  QUIT"
98  VTAB 22: GOSUB 28:E = 1: ON Z GOTO
       99,101,104,132,113,135,106,223:
       FLASH : PRINT : PRINT "TYPE A N
       UMBER FROM 1 TO 8": NORMAL : GO
       TO 98
99  HOME : PRINT  TAB( 9)"CATALOG OF DA
       TA FILES": PRINT :D = 0:W = 0:
       IF N = 0 THEN  GOSUB 3: GOTO 96
100  FOR I = 1 TO N: PRINT T$(I): GOSUB
       53: NEXT : GOSUB 50: GOTO 96
101  HOME : PRINT  TAB( 9)"** DESCRIBE
       FILE **": PRINT : GOSUB 57: IF
       P = 0 THEN 101
102  GOSUB 67: GOSUB 63: GOSUB 80: GOSU
       B 81: GOSUB 70: IF W = 0 THEN
       GOSUB 3
103  GOTO 96
104 E = 1: HOME : PRINT  TAB( 7)"** CRE
       ATE A NEW FILE **": PRINT :G =
       2: GOSUB 57: IF P = 1 THEN  GOS
       UB 67: GOSUB 61
105  PRINT "NUMBER OF ITEMS PER RECORD?
       ";:C = 50: GOSUB 31:A = Z:R = 0
       : FOR I = 1 TO A: GOSUB 41: NEX
       T :B = 0: GOTO 115
106  HOME : PRINT  TAB( 18)"COPY": PRIN
       T : PRINT "OLD";: GOSUB 57: IF
       P = 0 THEN 106
107  GOSUB 67: PRINT :G = 7:B = 0: PRIN
       T "NEW";: GOSUB 57: IF P = 1 TH
       EN  GOSUB 61
108  IF B > 0 THEN 129
109  VTAB 24: FLASH : PRINT "CREATING F
       ILE "A$: NORMAL
110  PRINT D$"OPENDES "A$: PRINT D$"DEL
       ETEDES "A$: PRINT D$"OPENDES "A
       $: PRINT D$"WRITEDES "A$: PRINT

       A: PRINT R: FOR I = 1 TO A : PR
       INT L$(I): PRINT L%(I): NEXT :
       PRINT D$"CLOSE"
111  FOR I = 1 TO N: IF A$ < > T$(I) TH
       EN  NEXT :N = N + 1:T$(N) = A$
       :F = 1
112  GOSUB 64: GOTO 96
113 G = 1: HOME : PRINT  TAB( 10)"## ED
       IT DESCRIPTION ##": GOSUB 57: I
       F P = 0 THEN 113
114  GOSUB 67:P = 3: GOSUB 61: PRINT :
       IF B > 0 THEN FOR I = 1 TO A:D$
       (I) = L$(I):S%(I) = L%(I): NEXT
115 R$ = "SRADINL":P = 39: GOSUB 82: PR
       INT  TAB( 3)"SAVE, REVIEW, ADD,
       DELETE, INSERT,": PRINT  TAB( 3
       )"CHANGE NAME, OR CHANGE LENGTH
       ......"
116 E=G: PRINT "(S, R, A#, D#, I#, N#,
       OR L#)";: GOSUB 4:E = 8:I = 1
117  IF  LEFT$ (Z$,1) =  MID$ (R$,I,1)
       OR I > 7 THEN 119
118 I = I + 1: GOTO 117
119  ON I GOTO 108,120,121,123,126,127,
       128: FLASH : VTAB 23: PRINT "HE
       Y!";: NORMAL : GOTO 116
120  GOSUB 70: GOTO 116
121  GOSUB 39: PRINT : PRINT "ADD "I" I
       TEMS:":Y = A + I: IF Y > 50 THE
       N  PRINT "YOU CAN'T HAVE MORE T
       HAN 50 ITEMS!": GOTO 116
122  FOR I = A + 1 TO Y: GOSUB 41:A = I
       : NEXT : GOTO 116
123  GOSUB 38: PRINT : PRINT : PRINT "D
       ELETE: "L$(I) TAB( 28)"LENGTH:
       "L%(I):R =R - L%(I) - 1: IF (I
       = A) THEN 125
124  FOR J = I TO A - 1:L$(J) = L$(J +
       1):L%(J) = L%(J + 1): NEXT
125 A = A - 1: GOTO 116
126  GOSUB 38:M$ = L$(I):Y = L%(I): GOS
       UB 41:A = A + 1: FOR J = A TO I
       + 2 STEP  -1:L$(J) = L$(J - 1):
       L%(J) = L%(J - 1): NEXT : L$(I
       + 1) = M$:L%(I + 1) = Y: GOTO 1
       16
127  GOSUB 38: PRINT "OLD NAME: "L$(I):
       PRINT : PRINT "NEW NAME?";: GOS
       UB 34: GOSUB 85:L$(I) = Z$: GOT
       O 116
128  GOSUB 38: PRINT L$(I)"  >>>  LENGT
       H IS "L%(I): PRINT : PRINT "NEW
       LENGTH?";:C = 115: GOSUB 31:R =
       R - L%(I) + Z:L%(I) = Z: GOTO 1
       16
129  HOME : FLASH : PRINT "REWRITING":
       NORMAL : FOR I = 1 TO A:S%(100
       + I) = 0: FOR J = 1 TO V: IF D$
       (J) = L$(I) THEN S%(100 + I) =
       J: GOTO 131
130  NEXT J
131  NEXT I:D$(0) = "*": PRINT D$"OPENA
       "D$",L"R: FOR Q = 1 TO B: PRINT
       D$"READ"A$",R"Q: FOR J = 1 TO V
       : INPUT D$(J): NEXT : FOR J = 1
       TO A:R$(J) =  LEFT$ (D$(S%(100
       + J)),L%(J)): NEXT : GOSUB 49:
       NEXT : PRINT D$"DELETE"A$: PRIN
       T D$"RENAMEA"D$","A$: GOTO 110
132  HOME : PRINT : PRINT  TAB( 9);: FL
       ASH : PRINT "###";: NORMAL : PR
       INT "  DELETE FILE  ";: FLASH :
       PRINT "###": NORMAL : VTAB 8: G
       OSUB 57: IF P = 0 THEN 132
133  IF I < > N THEN  FOR J = I TO N -
       1:T$(J) = T$(J + 1): NEXT
134  PRINT D$"OPEN"A$: PRINT D$"DELETE"
       A$: PRINT D$"OPENDES "A$: PRINT
       D$"DELETEDES "A$:F = 1:N = N -
       1: GOTO 96
135  HOME : PRINT : PRINT  TAB( 10)"EMP
       TY A FILE": GOSUB 57: IF P = 0
```

```
        THEN 135
136  PRINT D$"OPEN"A$: PRINT D$"DELETE"
        A$: GOSUB 67:B = 0: GOSUB 64: G
        OTO 96
137  PRINT : PRINT D$"PR#0": TEXT : HOM
        E : VTAB 2:E = 7: PRINT  SPC( 1
        1)"<<<  RECORDS  >>>": VTAB 8:
        PRINT "1 ENTER RECORDS" SPC( 6)
        "4 PRINT RECORDS": PRINT : PRIN
        T "2 DELETE RECORDS" SPC( 5)"5
        SORT": PRINT : PRINT "3 CHANGE
        RECORDS" SPC( 5)"6  QUIT"
138  VTAB 22: GOSUB 28: PRINT :E =5: ON
        Z GOTO 147,139,151,159,161,223:
        FLASH : PRINT : "TYPE A NUMBER
        FROM 1 TO 6!": NORMAL : GOTO 138
139 E = 9: GOSUB 83:I = 0
140  GOSUB 43: PRINT : PRINT "DELETED:
        ";: FOR J = 1 TO A: PRINT R$(J)
        " ";: NEXT : PRINT : PRINT "IS
        THIS WHAT YOU WANT DELETED?"
141  PRINT "(Y OR N)";: GOSUB 4: IF Z$
        = "Y" THEN  PRINT D$"WRITE"A$",
        R"I: PRINT  CHR$ (5): PRINT D$:
        GOTO 140
142  IF Z$ = "N" THEN 140
143  FLASH : PRINT "HEY!";: NORMAL : GO
        TO 141
144  HOME : HTAB 12: FLASH : PRINT "REW
        RITING": NORMAL : PRINT D$"OPEN
        A"D$",L"R:K = B:B = 0: FOR I =
        1 TO K: GOSUB 47: IF  LEFT$ (R$
        (1),1) < >  CHR$ (5) THEN B = B
        + 1:Q = B: GOSUB 49:D$(B) = R$(
        1)
145  NEXT
146  PRINT D$"DELETE"A$: PRINT D$"RENAM
        EA"D$","A$: GOSUB 64: GOTO 137
147  GOSUB 83
148 V = 5:0 = 1:K = A: FOR I = 1 TO A:R
        $(I) = "*": NEXT : I = B + 1
149  GOSUB 8: IF I < B + 1 THEN I = I +
        1: GOSUB 47: GOTO 149
150 B = B + 1: GOTO 148
151  GOSUB 83
152 E = 5: PRINT : PRINT "STARTING ITEM
        NUMBER? (DEFAULT= 2)";: GOSUB 4
        :E = 6: IF Z$ = "" THEN O = 2:
        GOTO 154
153 C = A: GOSUB 32:O = Z

154  PRINT - PRINT "ENDING ITEM NUMBER?
        ": PRINT "(DEFAULT= LAST ONE)";
        : GOSUB 4: IF Z$ = "" THEN K =
        A: GOTO 156
155 K = Z:C = A: GOSUB 32: IF K <
        O THEN PRINT : PRINT "THE LAST
        ITEM NUMBER MUST BE AT LEAST "O
        : GOTO 154
156 E = 6: PRINT : PRINT "STARTING ID?
        (DEFAULT= FIRST ONE)";: GOSUB 4
        :E = 10: IF Z$ = "" THEN X = 1:
        GOTO 158
157 J = LEN (Z$): FOR X = 1 TO B: IF  L
        EFT$ (D$(X),J) < > Z$ THEN  NEX
        T : PRINT : PRINT "ID NOT FOUND
        , TRY ANOTHER!": GOTO 156
158 V = 6: FOR I = X TO B: GOSUB 47: VT
        AB 6: CALL  - 958: PRINT "*ID=
        "D$(I): POKE 34, PEEK (37) + 1:
        GOSUB 8: NEXT I: GOSUB 64: GOTO
        137
159  GOSUB 83: GOSUB 80:I = 0
160  PRINT D$"PR#0": GOSUB 43: GOSUB 81
        : PRINT : PRINT :Z =0:D =0: FOR
        Q = 1 TO A: GOSUB 55: NEXT : GO
        TO 160
161  GOSUB 83: PRINT "NUMBER OF ITEM TO
        SORT BY?";: GOSUB 30: PRINT : P
        RINT "I WILL SORT BY "L$(Z): PR
        INT :V = Z - 1: PRINT : PRINT "
        ASCENDING (1)": PRINT "OR DESCE
        NDING (2) ORDER?";:C = 2: GOSUB
        31
162  HOME : FLASH : PRINT "SORTING": NO
        RMAL : FOR I = 1 TO B:S%(I) = I
        : PRINT D$"READ"A$",R"I: PRINT
        D$"POSITION"A$",R"V: PRINT D$"R
        EAD"A$: INPUT D$(I): NEXT : FOR
        I = B TO 2 STEP  - 1:K = 0: FOR
        J = 2 TO I: IF Z = 1 AND D$(J -
        1) > D$(J) THEN  GOSUB 56
163  IF Z = 2 AND D$(J) > D$(J - 1) THE
        N  GOSUB 56
164  NEXT : IF K THEN  NEXT
165  PRINT D$"OPENA"D$",L"R: FOR Q = 1
        TO B:I = S%(Q): GOSUB 47: GOSUB
        49:D$(Q) = R$(1): NEXT : GOTO 1
        46
166  PRINT : PRINT D$"PR#0": TEXT :E =
        7: HOME : HTAB 15: PRINT "+ REP
        ORTS +": GOSUB 84: PRINT : HTAB
        9: PRINT "1  RETRIEVE": PRINT :
        HTAB 9: PRINT "2  WITH SUMS": P
        RINT : HTAB 9: PRINT "3  FREQUE
        NCIES": PRINT : HTAB 9: PRINT "
        4 QUIT"
167  VTAB 21: GOSUB 28:C = 4: PRINT : G
        OSUB 32:S = Z: IF Z = 4 THEN 22
        3
168  HOME :E = 3: IF Z = 3 THEN 199
169 X = 0: PRINT "REPORT TITLE?";: GOSU
        B 4:M$ =Z$: PRINT "HOW MANY ITE
        MS TO PRINT?": PRINT "(DEFAULT
        = ALL)";: GOSUB 4: IF Z$ < > ""
        THEN 171
170 Y = A: FOR I = 1 TO Y:S%(I) = I:S(I
        ) = 0: GOSUB 74: NEXT : GOTO 172

171 C = A: GOSUB 32:Y = Z: PRINT "TYPE I
        N ITEM NUMBERS ONE AT A TIME.":
        PRINT : FOR I = 1 TO Y: CALL  -
        868: PRINT  SPC( 16)"<-";: HTAB
        13:F =  PEEK (37): GOSUB 30: VT
        AB F + 1: HTAB 1: CALL  - 868:
        HTAB 7: PRINT L$(Z):S(I) = 0:S%
        (I) = Z: GOSUB 74: NEXT
172 P = 40: GOSUB 82: HTAB 13: PRINT "S
        ELECT CASES":E = 3: PRINT : PRI
        NT "TYPE AN ITEM NUMBER,THEN IT
        S MINIMUM ANDMAXIMUM VALUES (IN
        THAT ORDER!). PRESS  RETURN AFT
        ER EACH  ONE.": PRINT : PRINT "
        TYPE A SLASH (/) TO SKIP THE 'O
```

```
       R'."
173  PRINT "TO IGNORE A MINIMUM OR A MA
       XIMUM, MERELYPRESS RETURN.": PR
       INT : PRINT "TO FINISH, TYPE A
       PERIOD (.), THEN PRESS  RETURN.
       ":E = 3:O = 0: PRINT
174 O = O + 1: PRINT : FOR K = 1 TO 5:U
       = (O - 1) * 5 + K: PRINT O"." S
       PC( 3)"ITEM NUMBER?";: GOSUB 4:
       E = 11: IF Z$ = "." THEN C = 2:
       ON K GOTO 186: GOSUB 73: GOTO 1
       88
175  IF Z$ = "/" AND K > 1 THEN C = 3:
       GOSUB 73: GOSUB 72: GOTO 184
176 C = A:W = 0: GOSUB 32:S%(200 + U) =
       Z:L = L%(Z): VTAB 23 - W: HTAB
       5: CALL  - 958: PRINT "** "L$(Z)
177  PRINT  TAB( 7)"MINIMUM?";: GOSUB 4
       : GOSUB 36: IF D = 1 THEN  PRIN
       T : GOTO 177
178 M$ (U) = Z$: IF Z$ = "." THEN C = 3
       : ON K GOTO 186: GOSUB 73: GOTO
       188
179  PRINT  TAB( 7)"MAXIMUM?";: GOSUB 4
       : GOSUB 36: IF D = 1 THEN PRINT
       : GOTO 179
180 N$(U) = Z$: IF Z$ = "" OR Z$ = "."
       THEN N$(U) =  CHR$ (95)
181  IF Z$ = "." THEN U = U + 1: ON K G
       OTO 188,188,188,188,189
182  IF K < 5 THEN PRINT : PRINT TAB( 8
       )"- OR -": PRINT
183  NEXT
184  IF O < 5 THEN PRINT : PRINT TAB( 8
       )"+ AND +": GOTO 174
185  IF O = 5 THEN 189
186 O = O - 1: IF O = 0 THEN C = 1
187  GOSUB 73: GOTO 189
188  GOSUB 72
189  IF S = 3 THEN 200
190  GOSUB 80:V = 40: IF W > 0 THEN PRI
       NT "HOW MANY COLUMNS PER LINE?"
       ;: GOSUB 4:V =Z
191  GOSUB 81: PRINT : PRINT : PRINT :
       PRINT  SPC( 10)M$: PRINT: PRINT
       :PRINT :D = 0:I = 0: IF X < V T
       HEN P = X: FOR J = 1 TO Y: PRIN
       T L$(S%(J)) SPC( 2 + S%(400 + J
       ) -  LEN (L$(S%(J))));: NEXT :
       PRINT : GOSUB 82
192 I = I + 1: GOSUB 76: IF G = 5 THEN
       195
193  IF X < V THEN  FOR K = 1 TO Y:Q =
       S%(K): PRINT R$(Q); SPC( 2 + S%
       (400 + K) -  LEN (R$(Q)));:S(K)
       = S(K) + VAL (R$(Q)): NEXT : GOS
       UB 52: GOTO 195
194  GOSUB 52: GOSUB 52: GOSUB 52: FOR
       K = 1 TO Y:Q = S%(K): GOSUB 55:
       S(K) = S(K) + VAL (R$(Q)): NEXT
195  IF I < B THEN 192
196  IF X < V AND S = 2 THEN  GOSUB 82:
       GOSUB 53: GOSUB 52: FOR K = 1 T
       O Y:G = INT (S(K) * 100 + .5) /
       100: PRINT G; SPC( S%(40  0 + K
       ) + 2 - LEN ( STR$ (G)));: NEXT
       : GOSUB 52: GOTO 205
197  IF S = 2 THEN GOSUB 52: GOSUB 52:
       GOSUB 52: PRINT SPC( 10)"SUMS:"
       : GOSUB 53:GOSUB 52: GOSUB 52:
       FOR K = 1 TO Y:Q = S%(K) : PRIN
       T L$(Q)": " INT (S(K) * 100 + .
       5) / 100: GOSUB 53: NEXT : GOTO
       205
198  GOTO 205
199  PRINT : PRINT "ITEM TO COUNT?";: G
       OSUB 30:Y = Z: PRINT : PRINT A$
       " MUST BE SORTED": PRINT "BY ";
       : FLASH : PRINT L$(Y): NORMAL :
       GOTO 172
200  GOSUB 80: GOSUB 81:R$ = "":D = 0:C
       = 0:S%(I) = Y:X = 0: GOSUB 74:X
       = X + 5:P = X + 6: IF W = 0 THE
       N P = 40
201  PRINT : PRINT : PRINT L$(Y) SPC( X
       - 3 - LEN (L$(Y)))"FREQUENCY":
       GOSUB 82: PRINT: FOR I = 1 TO B
       : GOSUB 76: IF G = 5 THEN 204
202  IF R$(Y) < > R$ AND I < > 1 THEN P
       RINT R$ SPC( X -  LEN (R$))C:R$
       = R$(Y): GOSUB 53:C = 1: GOTO
       204
203 C = C + 1: IF I = 1 THEN R$ = R$(Y)
204  NEXT : PRINT R$ SPC( X - LEN (R$))
       C: GOSUB 53
205  GOSUB 50: IF O < 1 THEN 166
206  PRINT : PRINT : PRINT : FOR Q = 1
       TO 30: PRINT "*";: NEXT : PRINT
       : PRINT "THESE CRITERIA WERE US
       ED:": PRINT : PRINT :I = 0
207 I = I + 1: FOR K = 1 TO 5:U = (I -
       1) * 5 + K:Q = S%(200 + U): PRI
       NT L$(Q): PRINT SPC( 5)"MIN:"M$
       (U): PRINT SPC( 5)"MAX:"N$(U):D
       = 14: GOSUB 52: IF S%(201 + U)
       = 0 THEN 210
208  IF K < 5 THEN  PRINT : PRINT  SPC(
       8)"- OR -": PRINT
209  NEXT K
210  IF I < O THEN PRINT : PRINT  "----
       - AND": PRINT : GOTO 207
211  GOTO 166
212  HOME :E = 7: PRINT  TAB( 15)"UTILI
       TIES": VTAB 7: HTAB 9: PRINT "1
       UPLOAD": PRINT : HTAB 9: PRINT
       "2 DOWNLOAD": PRINT : HTAB 9: P
       RINT "3 DRIVE SELECT": PRINT :

       HTAB 9: PRINT "4 QUIT"
213  VTAB 18: GOSUB 28:C = 4: PRINT : G
       OSUB 32: ON Z GOTO 214,216,218,
       223
214  GOSUB 84: GOSUB 65: FOR I = 1 TO B
       : GOSUB 47: FOR J = 1 TO A: IF
       LEN (R$(J)) < L%(J) THEN FOR K
       = 1 TO L%(J) - LEN (R$(J)):R$(J
       ) = R$(J) + " ": NEXT
215  NEXT :Q = I: GOSUB 48: NEXT : GOSU
       B 65: GOTO 91
216  GOSUB 84: GOSUB 65: FOR I = 1 TO B
       : GOSUB 47: FOR J = 1 TO A: FOR
       K = L%(J) TO 1 STEP  - 1: IF MI
       D$ (R$(J),K,1) < > " " THEN R$(
        J) = LEFT$ (R$(J),K):K=1
217  NEXT : NEXT :Q = I: GOSUB 48: NEXT
        : GOSUB 65: GOTO 91
218  HOME : VTAB 6:E = 4: PRINT TAB( 5)
        "DISK DRIVE FOR DATA FILES?";:C
        = 2: GOSUB 31:V = Z:N = 0: GOTO
        89
219 Y =  PEEK (222): IF Y = 255 THEN 223
220  IF Y = 5 THEN HOME : VTAB 10: PRIN
        T A$" IS EMPTY!": GOSUB 3: CALL
        768: GOTO 91
221  TEXT : IF Y = 6 OR Y = 11 OR Y = 2
        OR Y = 3 THEN CALL 768: GOTO 93
222  POKE 216,0: RESUME
223  IF F = 1 THEN GOSUB 66
224  TEXT : CALL  - 868: HTAB 17: FLASH
        : PRINT "SO LONG": NORMAL
225  DATA 104,168,104,162,223,154,72,15
        2,72,96,ENTER, DELETE, CHANGE,
        PRINT, SORT
```
©

WE REGRET—

WE CAN NO LONGER LIST ALL OF OUR PRICES

(that's how close we cut our margins).

But we still have great deals:

COMMODORE 8032
$1065.00

And we now have a mega-value catalog for just a dollar. Call or write today.

PHILADELPHIA COMPUTER DISCOUNT

P.O. Box 170
St. Davids, PA 19087

(215) 687-8540

MAIL ORDER PHONE

1-800-345-1289

If you have an Original or an Upgrade BASIC PET, you'll find this repeating-keys program frequently useful, especially when you need to make corrections to a large program.

PET Auto Repeat

Art Hunkins
School of Music
University of North Carolina at Greensboro
Greensboro, NC

These programs were adapted from several sources. SYS889 enables them both. The same command also *disables* the repeat function, which, incidentally, works for *all* keys. Auto repeat must be disabled for cassette functions to operate.

In the Original ROM version, POKE914,(30) specifies the .5 second delay time (hold time) before the character begins to repeat. POKE932, (4) specifies the repeat rate. Either value can be changed. For Upgrade ROMs, the locations are, respectively, POKE927,(30) and POKE945,(4).

Both programs store in the second cassette buffer. Whenever the need for extensive program editing arises, an auto repeat function is a real timesaver.

Program 1.

```
100 REM FOR ORIGINAL ROMS
110 DATA120,56,169,233,237,26,2,141,26
120 DATA2,88,96,173,3,2,201,255
130 DATA208,12,169,0,141,119,3,169
140 DATA30,141,120,3,208,30,238,119
150 DATA3,173,120,3,205,119,3,176,19
160 DATA169,4,141,120,3,169,0,141,3,2
170 DATA141,119,3,169,2,141,37,2,24
180 DATA76,133,230
190 FORI=889TO952:READJ:POKEI,J:NEXT
```

Program 2.

```
100 REM FOR UPGRADE ROMS
110 DATA120,56,169,233,229,145,133,145
120 DATA165,144,201,46,208,6,169,147
130 DATA133,144,208,4,169,46,133,144
140 DATA88,96,165,151,201,255
150 DATA208,12,169,0,141,119,3,169
160 DATA30,141,120,3,208,28,238,119
170 DATA3,173,120,3,205,119,3,176,17
180 DATA169,4,141,120,3,169,0,133,151
190 DATA141,119,3,169,2,133,168,24
200 DATA76,46,230
210 FORI=889TO963:READJ:POKEI,J:NEXT
```
©

We could tell you we make
We don't have to.

All for the Apple and Atari on diskette.

You keep Brøderbund at the top of the best seller lists, so you must know what it takes to make a great game. We think our APPLE and ATARI games are uniquely challenging with action and graphics second to none. But you keep asking for Brøderbund games because they're just plain fun.

Enjoy the magic of our growing collection. Lead a daring rescue mission in CHOPLIFTER. Or survive the maze and monsters in slithery SERPENTINE and LABYRINTH. Blast through the future with STAR BLAZER or conquer the ultimate pinball game with DAVID'S MIDNIGHT MAGIC. Take up train robbing in TRACK ATTACK or tense your mind and reflexes in DUELING DIGITS.

Brøderbund

Brøderbund products are available at your retailer or by writing to:
Brøderbund Software, Inc., 1938 Fourth Street, San Rafael, CA 94901 • Tel: (415) 456-6424

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

The Atari and Apple produce sounds of greater purity than some synthesizers, according to Frank Serafine, one of the creators of the TRON *sound track. He used these computers to create some of the movie's most memorable sound effects.*

The Sounds Of TRON

Tom R. Halfhill
Features Editor

His eyes locked in the grim intensity of combat, the video warrior takes aim at his opponent, slowly winds up like a baseball pitcher, and violently unleashes a glowing discus. Whooshing and screeching, the disc arcs over the arena toward its frantically dodging target – who has dodged too late. The disc demolishes the enemy gladiator in a split-second burst of light and sound.

The victorious warrior reaches up to retrieve his deadly disc, which zooms back to him like a boomerang. Standing tall in his glowing armor, TRON has triumphed again over the forces of the sinister Master Control Program.

This early scene from Walt Disney Productions' summer release, *TRON*, is typical of the film's pioneering use of animated computer graphics. But while *TRON*'s stunning visual effects have attracted the most attention from critics and audiences, fewer people are aware that the film's sound effects break new ground, too. The sounds, like the visuals, also are the product of high-technology computerization.

Apple And Atari Sounds

What is even more interesting, at least for personal computer enthusiasts, is the equipment used to fashion those sound effects: many were generated with an off-the-shelf Atari 800 and an Apple II.

Not only that, but an Atari 800 running a commonly available data base manager program was used to store, categorize, index, and instantly retrieve all the special sound effects collected for *TRON*. How many was that? "Oh, my gosh – thousands, just thousands of sound effects," says one of their key creators, Frank Serafine, of Serafine FX Music/Sound Design.

SFX – you can see its credits roll by at the end of *TRON*, if you watch closely – is a Los Angeles-based sound design studio hired by Disney to collect and create most of the special sounds heard in *TRON*. Equipped with the most advanced audio components and sound-dedicated computers available, SFX previously had done sound effects for *Star Trek: The Motion Picture* and *The Fog*. But *TRON* was the studio's most innovative and involved project by far – SFX labored for a year and three months shaping the film's sounds.

Electronic Sound Assembly

Serafine says the job would have taken even longer had he used the established method, improved little since the 1930s, of cutting and splicing bits and pieces of the soundtrack on mechanical film editors. Faced with a staggering task, Serafine turned to computers.

With his broad array of audio and computer equipment, including the Atari and Apple, Serafine designed a process he calls "Electronic Sound Assembly." Used for the first time in *TRON*, it does for the creation of sound what word processing does for writing: it allows the manipulation and fine-tuning of the work on a video screen. Serafine can digitalize sound effects, feed them into a Fairlight CMI (Computer Musical Instrument), plot the waveforms on a monitor and tinker with them almost endlessly. In fact, he can actually alter the sound directly on the screen with a light pen.

The results are sounds honed to an unbelievable degree of detail. Serafine says he was inspired by the sound-layering techniques pioneered by the Beatles in the 1960s with considerably less sophisticated equipment. "They achieved a subliminal effect, something which made you want to listen to their music over and over again to hear every sound. That's what I tried to do with the sound effects in *TRON*."

Consider the roar of the "light cycles," the futuristic motorcycles on which the gladiators of *TRON* duel while racing along the circuit grids of the computer in which they are trapped. To make the light cycles seem real, Serafine assembled and combined more than 50 different sounds. When a cycle makes a 90-degree turn to cut off an opponent, for example, the sound effect is a combination of video game tones generated on the Atari and a recording of a buzz saw.

Likewise, the sound of the sleek tanks which prowl the circuit canyons in *TRON* are a compilation of dozens of noises made with the Atari, all layered together. Serafine first tried recording a real army tank, but was disappointed by the clanking rattle. "I wanted something that sounded more turbine-like, more computer-controlled," he explains.

Screaming Monkeys

Other notable sound effects in *TRON* generated by

Serafine in his studio, flanked by his Atari and Apple.

the Atari include numerous video game bleeps and zaps, the sound of the "grid bugs" which arise from the circuits below the "solar sailer," the shock prods wielded by the Master Control Program's guards, and the collisions of tanks with walls (combined with recordings of real military explosions).

The Apple II was used with plug-in sound cards from Mountain Hardware and an Alpha Centauri keyboard. These add-ons made the Apple capable of a wider range of sound effects than the Atari, says Serafine, but also made it harder to use. Still, the added capability was an advantage when programming certain video game sounds and the "bonging" noise of a thrown discus.

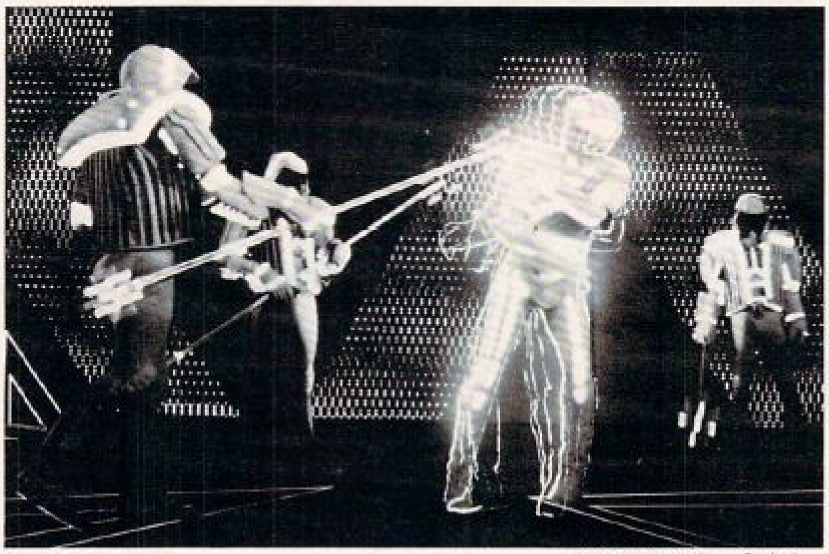
The discus sound is a good example of how much work went into each effect. The Apple's "bonging" sound was overdubbed with recordings of a bullwhip and of monkeys screaming at the San Diego Zoo.

"The director [Steven Lisberger] demanded a concept for each sound effect," explains Serafine. "I couldn't just go around doodling with sound effects. We had to sit around with the director in discussion sessions to talk over the concept of each sound effect. Like, for the disc-throwing sound, we came up with the concept that they had to sound beautiful, yet sad – sad because something so beautiful can at the same time kill. So overlaying the monkey screams lets you know that, although this flying disc is really beautiful, you also know you'd hate to be hit by it."

It was originally Disney's idea to involve personal computers in the sound production. Serafine's background is in audio and multi-media presentations, not computers. He was designing planetarium shows in Colorado in 1976 when he first attracted Disney's attention. Disney hired him to put together a multi-media presentation for the grand opening of Space Mountain at Disneyland in 1977.

A few years later, after Serafine founded SFX, Disney hired him to create the sound effects for *TRON* and sent him to "Silicon Valley" in California, the home of America's microcomputer industry. Disney figured it was the ideal place to find video game sound effects for a movie whose central theme was to be video games. Serafine met with representatives from Apple and Atari, who set him

Sark, assistant to the arch-villain Master Control Program, introduces the killer disc.



© MCMLXXXII Walt Disney Productions.

Crack Programmer Flynn finds himself at the mercy of distinctly user-unfriendly guard programs.

up with computer systems. Atari also lent him an Atari Sound Development Disk, a well-guarded, powerful utility package rarely entrusted to anyone outside Atari itself.

Most of the Atari sounds were not actually programmed by Serafine, though. He got help from Ed Rotberg, an ace Atari programmer who has since left Atari to form his own company, and from another expert – Laurent Basset, who is 17. Basset is the son of one of Serafine's friends.

"He's a whiz kid," says Serafine. "This kid was actually able to do anything I wanted done on the machine. I would dream of a sound or a concept, and he would come back to me the next day with the finished programs."

All the thousands of sound effects created on the computers or collected on tape were cataloged on the Atari with *FileManager 800*, a data base program by Synapse Software. Serafine says it saved his studio hours of tedious filing. The record for any sound effect, listing its characteristics, source, and location on tape, could be retrieved in 1.5 seconds.

A Clean, Pure Sound

Serafine, who had only limited previous experience with personal microcomputers, was also impressed with the sound quality of the machines. "The amazing thing to me is the purity of the sound that comes out of the Atari, and also the Apple. Their sound chips produce,an extremely clean, pure sound which is even superior to some synthesizers I've worked with. We had no trouble using those sounds in the movie."

In fact, he thinks the sound capabilities of the computers are underused, partly because the proper tools are not available. "I think Atari and Apple will look at the success of the personal computers [in *TRON*] and develop better sound development disks as a result. For example, Atari has a *Music Composer* which gives you all the tools you need to compose music except really good sound. You can't access all the good sounds in the machine with the *Music Composer*. Someone at Atari ought to combine the *Music Composer* with features of the Sound Development disk, and they'd really have something. Maybe if they read this...."

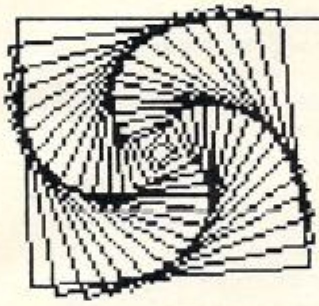
Serafine sees a future for personal computers in other productions involving video game-like effects. In fact, his current project is a *Pac-man* commercial for Seven-Up.

Unfortunately, most people take sound effects for granted, he says. The sounds in *TRON* come and go so fast that almost nobody realizes the amount of labor involved. "Several people spend a week of intense work to create something," he sighs, "and it lasts only one second."

Three Atari sound effects by Laurent Basset, who helped program many of the Atari sounds in *TRON*.

```
10 ? CHR$(125):POKE 710,80:POKE 755,0:? "
THUNDER & RAIN"
14 FOR TIME=0 TO 1
15 B=INT(255*RND(0)+50):X=RND(0)*200
20 FOR PITCH=1 TO B:SOUND 0,PITCH,8,15:NEXT PITCH
25 FOR T=1 TO X:NEXT T:NEXT TIME:SOUND 0,0,0,0
27 FOR RAIN=0 TO 15 STEP 0.2:SOUND
1,0,0,RAIN:FOR W=1 TO 20:NEXT W:NEXT RAIN:FOR
W=1 TO 2000:NEXT W:SOUND 1,0,0,0
40 FOR X=1 TO 100:NEXT X
50 ? CHR$(125):POKE 710,80:POKE 755,0:? "
HEART BEAT"
60 FOR Y=1 TO 5:FOR W=1 TO 40:SOUND
0,12,3,15:NEXT W:FOR W=1 TO 150:SOUND
0,0,0,0:NEXT W:NEXT Y:SOUND 0,0,0,0
70 FOR X=1 TO 200:NEXT X
80 ? CHR$(125):POKE 710,80:POKE 755,0:? "
STEAM LOCOMOTIVE"
90 X=0.1
100 FOR W=1 TO 150
110 FOR LOUD=15 TO 4 STEP -X:SOUND
0,15,0,LOUD:NEXT LOUD
120 X=X+0.01:IF X>0.7 THEN X=0.7
130 NEXT W:SOUND 0,0,0,0
135 FOR S=1 TO 2
140 SOUND 1,40,10,10:SOUND 2,10,10,8:SOUND
3,90,10,10
145 FOR X=1 TO 200:NEXT X
147 SOUND 1,0,0,0:SOUND 2,0,0,0:SOUND 3,0,0,0
148 FOR X=1 TO 60:NEXT X
150 NEXT S
```

A Monthly Column



Friends Of The Turtle

David D. Thornburg
Associate Editor

Battle Of The UFL's

The development of User Friendly Languages (UFL's) is proceeding so quickly that any report is likely to be outdated by the time it appears. Nonetheless, there is enough interest in the UFL's for the Atari, TI, and Apple computers to warrant an overview of the best offerings for these machines. The user friendly languages of principal interest seem to be PILOT and LOGO. It so happens that Atari PILOT (and Apple SuperPILOT) incorporate turtle graphics. While turtle graphics (common to all LOGO's) is not essential for a language to be user-friendly, it helps.

Rather than detail all UFL's for each computer, I will restrict the analysis to Atari PILOT, TI LOGO and Apple LOGO. The differences between Apple LOGO and the Apple versions of LOGO produced by Terrapin and Krell are deserving of separate comment later. (I have just received Krell LOGO and will need to use it some more before writing about it.)

What makes the following comparison interesting is the tremendous difference in price and features of the three chosen language systems. The table summarizes all three configurations. I have listed the bare minimum configuration needed to make the language work. If you want to save your programs, the cost of a recorder must be added to the Atari and TI systems. Since all three systems require a separate display, I have left that item out of the cost analysis.

The entries in this table reflect questions readers have been sending to Friends of the Turtle.

The Atari PILOT system is the least expensive. This results from the low cost of the computer and from the fact that Atari PILOT can be used with a minimum amount of RAM. The increased memory requirement of TI LOGO results in a profoundly increased cost for that system – a cost difference we would not have if we were comparing BASIC's. Since Apple LOGO requires both 64K of RAM and a disk drive, it is the most expensive of the systems. However, Apple LOGO is by far the most powerful of the languages under consideration.

The turtle graphics implementations are ex-

Table. System Comparison

Feature	Atari PILOT	TI LOGO	Apple LOGO
Minimum System	Atari 400 Pilot cartridge	TI 99/4A 32K memory exp. LOGO cartridge	Apple II or II+ language card floppy disk LOGO disk
List Price	$429	$980	$2625
Visible Turtle	No	Yes	Yes
Turtle Graphics Resolution	160 X 80	256 X 192 (may "run out of ink")	280 X 240 (vertical scale is changeable)
Number of Simultaneous Colors	4	16	6
Total Color Range	16 hues X 8 luminances	16	6
Character Font Editor	No	Yes	No
Multiple Dynamic Turtles	No	32	No
Real Number Arithmetic	No	No	Yes
Unlimited Tail-end Recursion	Yes	No	Yes
Recursion Depth Before Crash	8	end of memory	end of memory
Access to Joysticks, etc.	Yes	No	Yes
Full Stroke Keyboard	No	Yes	Yes
TV Sound Generator	Yes	Yes (in LOGO II)	No
Direct Memory Access	Yes	No	Yes

cellent in all three systems. Atari PILOT is the only one that does not have a visible turtle, but this can be remedied somewhat with the Visiturt program I published a few months ago (**COMPUTE!,** April 1982, #23). The Atari system has the lowest resolution, but has the greatest color accuracy and range of the three languages. A major annoyance with the TI system is the "out of ink" error that arises when trying to create complex pictures. Since TI creates high resolution graphics by dynamic character definition (a topic for a later column), it is not as versatile as a true memory mapped display. Multiple velocity turtles (turtles that have speeds as well as positions and orientations) are only available on TI LOGO. Up to 32 such animated characters can be created with any of 26 shapes formed in a 16 X 16 dot matrix. While the Atari hardware allows for such animated characters (called players), PILOT users must gain access to these through machine language instructions. Of the three systems, only TI allows the user to interactively modify or define the shapes of characters and velocity turtles.

If you are content with integer arithmetic, any of the systems will do. If you must have access to decimal fractions, only Apple LOGO will meet your needs. Interestingly enough, the restriction to integer arithmetic can result in minor graphics problems (drawing a regular seven-sided polygon, for example) in Atari PILOT and TI LOGO, although these problems can be easily overcome by careful programming.

Recursion is of two types. A simple jump to the beginning of a procedure is called tail-end recursion. Recursion involving the use of a procedure that ultimately returns to the calling procedure is more difficult since the computer must keep track of the sequence and names of all calling procedures. As a result, recursion can use up all free memory just by keeping track of this information. Atari PILOT allows unlimited tail-end recursion (as does Apple LOGO), but allows only eight nested procedure calls.

TI LOGO differs from both Atari PILOT and Apple LOGO in that the user is not provided with access to joysticks nor to the direct reading and alteration of memory. Both Atari PILOT and Apple LOGO have the equivalent of BASIC PEEK and POKE commands to allow the examination and alteration of the contents of arbitrary memory locations.

The keyboard quality is highest for the Apple II, although TI's decision to use a conventional keyboard makes that machine easy to use as well. My experience is that the Atari 400 membrane keyboard is acceptable to children, but is annoying to adults accustomed to typewriters. Since the Atari 800 has a fine full-stroke keyboard, this option is available to those willing to pay the higher price.

In summary, each system has strong features and drawbacks. You are certain to like some aspects of each system. For the price, the Atari PILOT system is beyond comparison. On the other hand, Apple LOGO is a powerhouse of a language, and its features are well worth its price. The ease with which animated sprites can be created and used in TI LOGO makes this system a natural choice for anyone interested in animation.

All three manufacturers are in this business for the long haul, so your selection should be based purely on needs and budget.

Apple LOGO And The Silentype Printer

Those of you who use the Silentype printer with your Apple computer have probably wondered how to get copies of the displays of turtle graphics created by LOGO procedures. The easiest way I have found is to initialize the printer before loading LOGO. When you initialize a file diskette, it contains a program named HELLO. Normally (for LOGO) there will be no statements in this program. However, if you were to boot this disk first rather than start with the LOGO disk, the HELLO program would be automatically run. Since your Apple already has one dialect of BASIC in ROM (either integer or Applesoft), then you could use a BASIC HELLO program to initialize the Silentype printer.

The Silentype manual shows the numerous ways in which the printer's graphic features can be set up. The default mode lets the printer print bidirectionally. One set of dots is printed as the head moves from left to right and the second set is drawn as it moves from right to left. While this significantly improves the printing speed, the slack in the Silentype mechanism causes this mode to produce unacceptable vertical misalignment when printing high resolution graphics. The unidirectional printing mode does not have this problem.

Second, the Silentype normally prints images just as they appear on the screen. If you have a few white lines on a black background, that is how the printer will print the picture. Normally one expects the reverse of this for line drawings – the background should be white and the lines should be dark. As a result, the printer needs to have its color fields reversed.

Both of these changes are made in the BASIC program shown below. This program assumes that the Silentype printer interface is located in slot #1 and that the disk drive is located in slot #6.

```
10 D$="":REM D$ CONTAINS CTRL-D

20 PRINT D$;"PR #1"
```

```
30 PRINT

40 POKE -12529,255

50 POKE -12524,0

60 PRINT D$;"PR #0"

70 PRINT "GRAPHICS PRINTER INITIALIZED"

80 PRINT "INSERT LOGO DISK AND PRESS RETURN"

90 INPUT A$

100 PRINT D$;"PR #6"

110 END
```

Once this program has been saved in the HELLO file, your printer will be automatically initialized. To set up the printer, you must first start the computer with the file diskette that has this HELLO program. Once the display instructs you to insert the LOGO disk, you should do that and press RETURN. Now you will have LOGO in the computer and also have a properly initialized printer.

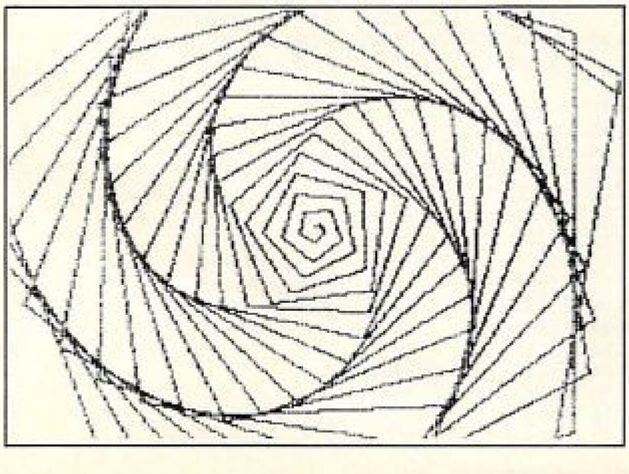
To print a high resolution screen image from LOGO you can use the procedure:

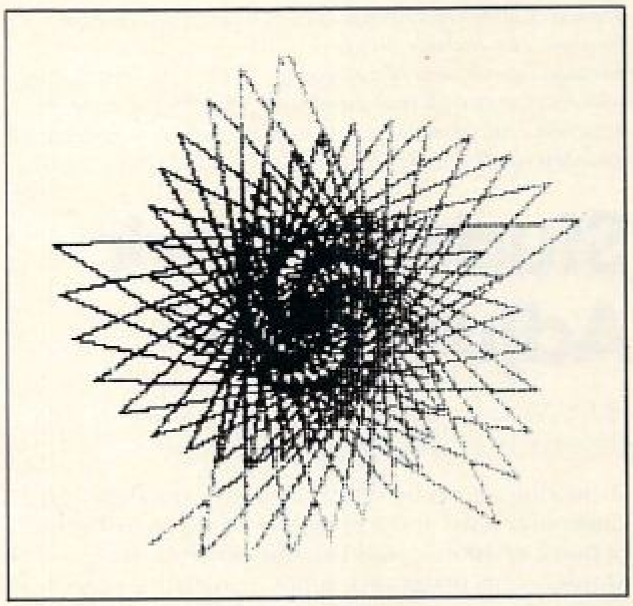
```
TO PICT
.PRINTER 1
PRINT CHAR 17
.PRINTER 0
END
```

From then on, any time you enter PICT the current graphics screen will be copied onto the printer.

If you try printing an image of a square or a circle, you may notice that the image is squashed vertically. This results from a difference in the aspect ratio of the printer and your TV display. To print pictures with a perfect aspect ratio, you must enter

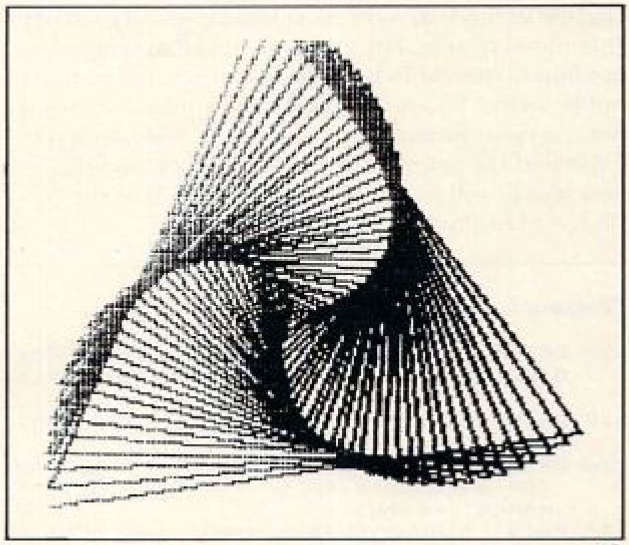
```
SETSCRUNCH 1
```





before drawing the figure you want to print. Once the aspect ratio has been changed to this value (from its default value of 0.8), all your pictures will come out perfectly. The accompanying figures show some of the results.

The Silentype printer is an excellent tool for capturing your LOGO graphic images. It is time you put it to work!



©

For Atari and Apple, this is the first of a three-part series on PILOT. This month, an Apple version (Microsoft BASIC) – and next month an Atari BASIC PILOT will be published in Part II. The Apple version requires Applesoft, 32K memory, and one disk drive.

Part I

Turtle PILOT

Alan W. Poole
Loomis, CA

How would you like a powerful new language for your computer that combines PILOT, turtle graphics, and all commands and functions? The programs at the end of this article create a version of PILOT which contains all of these features. Best of all, unlike most languages available for the Apple, this language isn't going to cost you a fortune. This version of PILOT, which I have named Turtle PILOT, has been patterned after Atari PILOT. Most Atari PILOT programs can be converted to Turtle PILOT without very many changes.

At the end of this article are two program listings. The Turtle PILOT Editor is used for typing PILOT programs. The Turtle PILOT Translator writes a program in Applesoft which is equivalent to a PILOT program typed with the Editor. Both programs require an Apple with Applesoft, 32K, and one disk drive. Also included is an example showing everything typed and printed on the screen while entering and translating a program, along with a sample RUN after it was translated.

Introduction To PILOT

PILOT is a simple language and is very easy to learn, especially if you already understand BASIC. In this article I will assume that the reader is familiar with Applesoft BASIC. To get an idea of how PILOT works, consider the following program and explanations for each line.

```
1  *QUIZ
2  T:HOW MUCH IS 8+4?
3  A:
4  M:12,TWELVE
5  TY:THAT'S CORRECT.
6  TN:NO, TRY AGAIN.
7  JN:*QUIZ.
```

The first line is a label, which is used to identify sections and modules (modules are subroutines) of a program. Labels always begin with an asterisk. In the second line the T is the instruction name for Type. Everything following the colon will be displayed on the screen. The third line Accepts a response from the user. Line four uses the Match instruction. Each item following the colon is compared with the last response. The Y in line five is the Yes conditioner. The Yes conditioner causes the instruction to be executed only if the last Match succeeded. The N in line six is the No conditioner. A line with a No conditioner will be executed only if the last Match failed. The last line causes a Jump back to the first line if the question was answered incorrectly. Notice that a label is used instead of a line number. The line numbers are not actually part of the program, but are used only to make editing easier.

Parts Of An Instruction

The order of the parts of an instruction is important. Below is a description of each of the elements in an instruction. Although an instruction does not have to contain all of the optional elements, the elements that are included must be in the order they are given below.

1. Instruction Name – The instruction name is a single letter and always comes first. It is required with every line other than a label, since labels are not considered instructions.

2. Conditioner – The conditioner is either a Y or an N. The conditioner is optional and may be used with any instruction.

3. Expression – An instruction with an expression is executed only when the expression is true. The expression must be placed between parentheses. Expressions are optional and may be used with any instruction. Below are some examples of instructions that include expressions.

```
T(S>100):VERY GOOD.
J(L=SQR(N)):*START
AY(X=0 AND MID$(I$,M,1)=STR$(J)):
```

4. Colon – Every instruction must have a colon.

5. Object – The last part of an instruction is the object. Everything following the colon is called the object of the instruction. An object is optional with some instructions.

The Turtle PILOT Instructions

There are 11 instructions in Turtle PILOT, not including the turtle graphics commands. Below is an explanation of each instruction. At the end of each explanation are a few samples of the instruction.

T: Type. The Type instruction will print everything in the object on the screen. The Type instruction has an advantage over BASIC's PRINT command. With the Type instruction, words at the end of a line will not usually be divided between two lines of the screen. A string variable can be Typed by placing the name of the string variable in the object preceded and followed by a pound sign (#). An ampersand (&) placed at the end of the object will cause the next printed character to continue on the same line. The ampersand will not be displayed on the screen. A Type instruction without an object will print a blank line.

```
T:
T:HELLO,$NAME$ &
TY:YOUR TOTAL SCORE IS #S#.
```

A: Accept. The Accept instruction inputs a response from the user. The Accept instruction is very similar to BASIC's INPUT command, with the advantage that any character can be typed without an error occurring. An object is not required, but the input can be assigned to a variable by placing the variable name in the object.

```
A:
A:NAME$
A(M(K)=0):X
```

M: Match. The Match instruction compares all of the items in the object to the last response. The items in the object are separated by commas and may include string variable names. If any of the items match with the last response, the Y conditioner is set. Otherwise, the N conditioner is set. The Match instruction does not compare the item with just the start of the last response. It searches for the word through the entire response. For instance, GO would Match with GOING, INGOT, and LINGO. You may put up to 25 items in the object.

```
M:HI
M:A,E,I,O,U,Y,A$,B$
MN(Z<20):END,STOP
```

J: Jump. The Jump instruction causes a branch to the line with the label that matches the label in the object of the Jump instruction. This instruction resembles BASIC's GOTO command, except a label is used instead of a line number.

```
J:*START
JY:*PART TWO
```

U: Use. The Use instruction is for Using modules (subroutines) in a PILOT program. It is similar to the Jump instruction, but the computer remembers the line from which it came. Program execution will continue at the line with a label that matches the label in the object until an End instruction is encountered. The End instruction will cause a return to the line following the Use instruction. The Use instruction is like BASIC's GOSUB command, with labels used instead of line numbers.

```
U:*PRINT
UY:*FIRST
```

E: End. The End instruction will terminate the program unless a Use instruction has been executed. If a Use instruction has been executed, program execution will continue at the line following the Use instruction. No object is used with an End instruction. The End instruction is similar to BASIC's RETURN and END commands.

```
E:
E(N=T):
```

R: Remark. The Remark instruction is not executed and is used only for program documentation.

```
R:THIS IS A REMARK
```

C: Compute. The Compute instruction may be used for numeric calculations or string manipulation.

```
C:N=N+1
C:S(K)=SIN(A*10)
CY:A$="ABC"
C(T=1):Z$(N,1)=X$+RIGHT$(I$,3)
```

B: Basic. The object of the Basic instruction may contain any Applesoft commands.

```
B:HOME
B:GET K$:PRINT K$;
BY:HPLOT 10,Y TO X,50
BN(V>3):COLOR=2: HLIN 10,20 AT Y
```

S: Sound. The object of the Sound instruction should contain a number from 1-31 for the pitch of the note, a comma, and a number from 1-255 for the duration. The notes range from C below middle C for a pitch value of 1 to F# above C above middle C for a value of 31. These values are the same as the pitch values used in Atari PILOT.

```
S:10,200
S:30,D
SY:P,SQR(L*2)
```

G: Graphics. The Graphics instruction precedes all turtle graphics commands, which will be explained next month in Part II.

Variables In Turtle PILOT

You may use any variable in a Turtle PILOT program that you could use in an Applesoft program, except variables beginning with Q. The reason for not using Q variables will be explained next month in Part II of this article. You may also use any of Applesoft's mathematical and string functions.

Using The Editor

The Editor has 11 commands to help you in typing Turtle PILOT programs. The Editor has two modes. First there is the command mode, indicated by a prompt. Any of the Editor's 11 commands may be typed in the command mode. The second mode is the program mode, indicated by a line number that is automatically printed on the left side of the screen. The program mode may be entered through several of the Editor commands. PILOT programs are typed in the program mode. To return to the command mode from the program mode, press RETURN without typing anything. When you have finished using the Editor, press the ESC key. Accidentally pressing the ESC key can be corrected by pressing RETURN immediately. Following is a description of each of the Editor commands.

The ADD command is used for entering the program mode to start or continue a program. It will ADD lines to the end of the program. The ADD command may also be followed by a line number. The lines ADDed to the program will then start at the line number specified. All the lines of the program currently in memory from that line to the end of the program will be erased.

The LIST command will LIST the program in memory. To LIST a single line, type the line number following the LIST command. A range of lines can be LISTed by typing the first and last line numbers separated by a comma. All the lines of a module may be LISTed by specifying the label of the module. Pressing RETURN will abort a LISTing.

The EDIT command is used to change line(s) of a program. It may be followed by a single line number or a range of line numbers with the first and last line numbers separated by a comma. The line to be EDITed will appear on the screen with the cursor at the beginning of the line.

The INSERT command is used for adding a line in the interior of a program. The INSERT command must be followed by a line number. No lines of the program will be deleted, but the numbers of the lines after the INSERTed line will be raised by one.

The DELete command is used for erasing line(s) from a program. A single line, a range of lines, or a label of a module may be specified. The numbers of the lines following the DELeted lines will be lowered.

The NEW command erases the program in memory.

The LOAD command will read a PILOT program from the disk and append it to the program in memory. The NEW command must be used first if the program in memory is not wanted. The name of the program is specified by following the LOAD command by that name.

The SAVE command will store the program in memory on the disk under the name specified. A program must be SAVEd before it can be translated.

The MEM command prints the number of free bytes available.

The CATalog command will print a catalog of the files on the disk. Turtle PILOT programs will appear as text files with ".P" at the end of their names.

The PR# command changes output to the specified slot, allowing a printer to be used with the Editor.

Using The Translator

The Translator is used to translate your PILOT programs into Applesoft programs. When you RUN the Translator, it will ask you to type the name of the program to be translated. Make sure the program to be translated has been SAVEd on disk. After you have typed the name, the translating will automatically be done. The translated Applesoft program will be sent to a text file on the disk that can be EXECuted later to load it into memory. The computer will tell you what to type to load the program into memory. Once it is in memory, you can RUN, SAVE, LOAD, and LIST it just like a normal Applesoft program.

Errors

Major syntax errors will be caught by the Editor, and the Editor will have you retype the line. Most other errors will be detected by the Translator while the program is being translated. If an error occurs during a translation, the computer will print "ERROR IN PILOT LINE NO." followed by the number of the line in which the error occurred. The computer will then automatically RUN the Editor. You should LOAD the program you were translating, correct the error, SAVE the program, and RUN the Translator again. Since this is a time consuming process, you should look the program over carefully before trying to translate. If an error occurs while a translated program is RUNning, divide the line number by ten to calculate the corresponding PILOT line number.

Typing The Programs

A typing error in the Translator program could produce disastrous results. To make it easier to find mistakes, the following line should be included when the Translator is first typed.

```
15 POKE 216,0: GOTO 30
```

This will cause the translated program to be sent to the screen instead of the disk, and error messages will be printed normally. When all mistakes have been corrected, delete line 15.

The turtle PILOT Editor must be SAVEd under the name EDITOR, and the Turtle PILOT Translator must be SAVEd under the name TRANSLATOR. Make sure you SAVE the Translator immediately after typing it and before you RUN it, since it erases itself when it is finished.

This is only the first in a series of three articles about Turtle PILOT. In Part II we'll cover turtle graphics, which is probably the best feature of Turtle PILOT, and we'll take a look at some other features of the language. In Part III we'll convert an Atari PILOT program to Turtle PILOT. If you are already familiar with turtle graphics, experiment with the commands listed in line 10230 of the Translator program and see what you can discover.

The author has offered to make a copy of the programs for you. Send a blank disk (specify DOS 3.2 or 3.3), a stamped, self-addressed mailer, and a $3.00 copying fee to:

Alan Poole
4728 King Road
Loomis, CA 95650

Program 1.

```
1 REM  TURTLE PILOT TRANSLATOR
2 REM  BY ALLAN POOLE
1Ø GOSUB 1ØØØØ
17 REM
18 REM *** OPEN EXEC FILE ***
19 REM
2Ø N$ = LEFT$ (N$,LEN (N$) - 1) + "EXEC": PRI
    NT D$"OPEN"N$: PRINT D$"DELETE"N$
21 PRINT D$"OPEN"N$: PRINT D$"WRITE"N$
3Ø POKE 33,3Ø: LIST 5ØØØØ - 6ØØØØ: POKE 33,4Ø

4Ø PRINT "5 GOSUB 5ØØØØ"
47 REM
48 REM *** MAIN LOOP ***
49 REM
5Ø FOR LN = 1 TO NL:LN$ = ""
6Ø LN$ = LN$ + STR$ (LN * 1Ø)
7Ø I = Ø: FOR L = 1 TO 12: IF LEFT$ (P$(LN),1
    ) = I$(L) THEN I = L
8Ø NEXT : IF I = Ø THEN 15ØØØ
9Ø GOSUB 2ØØ: GOSUB 3ØØ: GOSUB 4ØØ
1ØØ ON I GOSUB 5ØØ,1ØØØ,15ØØ,2ØØØ,25ØØ,3ØØØ,35
    ØØ,4ØØØ,45ØØ,5ØØØ,6ØØØ,65ØØ
11Ø PRINT LN$: NEXT
117 REM
118 REM *** END OF PROGRAM ***
119 REM
12Ø PRINTLN*1Ø;"END":PRINT "?";CHR$(34);"YOUR ~
    TRANSLATED PROGRAM IS IN MEMORY";CHR$
    (34)
121 PRINT D$"CLOSE"
13Ø PRINT : PRINT "TO LOAD YOUR TRANSLATED PRO
    GRAM INTO":
131 PRINT "MEMORY,TYPE "CHR$ (34);"EXEC "N$; C
    HR$ (34)
14Ø NEW
197 REM
198 REM *** SPLIT PILOT LINE AT COLON ***
199 REM
2ØØ FOR L = 1 TO LEN (P$(LN)): IF MID$ (P$(LN)
    ,L,1) = ":" THEN T = L:L = 3ØØ
21Ø NEXT
215 IF LEFT$ (P$(LN),1) = "*" THEN L$ = "*":R$
     = P$(LN): RETURN
22Ø L$ = LEFT$ (P$(LN),T - 1): IF T = LEN (P$(
    LN)) THEN R$ = "": RETURN
23Ø R$ = RIGHT$ (P$(LN), LEN (P$(LN)) - T)
24Ø T$ = L$: GOSUB 11ØØØ:L$ = T$
25Ø IF LEFT$ (L$,1) = "G" THEN T$ = R$: GOSUB ~
    11ØØØ:R$ = T$
26Ø RETURN
297 REM
298 REM *** FIND CONDITIONER ***
299 REM
3ØØ C = Ø: IF LEN (L$) < 2 THEN RETURN
31Ø IF MID$ (L$,2,1) = "Y" THEN LN$ = LN$ + "I
    F QC=1 THEN ":C = 1
32Ø IF MID$ (L$,2,1) = "N" THEN LN$ = LN$ + "I
    F QC=Ø THEN ":C = 2
33Ø RETURN
397 REM
398 REM *** FIND EXPRESSION ***
399 REM
4ØØ EX$ = "": IF RIGHT$ (L$,1) < > ")" THEN RE
    TURN
41Ø T = Ø: FOR L = 1 TO LEN (L$) - 1: IF MID$ ~
    (L$,L,1) = "(" THEN T = L:L =3ØØ
42Ø NEXT :EX$ = MID$ (L$,T + 1, LEN (L$) - T -
     1):LN$ = LN$ + "IF"+EX$ + "THEN"
43Ø RETURN
497 REM
498 REM *** T: INSTRUCTION ***
499 REM
5ØØ LN$ = LN$ + "QT$=" + CHR$ (34): IF R$ = ""
     THEN LN$ = LN$ + CHR$ (34) + "
5Ø1 GOSUB 51ØØØ": RETURN
51Ø FOR L = 1 TO LEN (R$):T$ = MID$ (R$,L,1)
52Ø IF T$ = "$" THEN 6ØØ
53Ø IF T$ = "#" THEN 7ØØ
54Ø LN$ = LN$ + T$
55Ø NEXT :LN$ = LN$ + CHR$ (34) + ":GOSUB 51ØØ
    Ø": RETURN
6ØØ IF L > LEN (R$) - 2 THEN 54Ø

61Ø T = Ø: FOR L1 = L + 2 TO LEN (R$): IF MID$
     (R$,L1,1) = "$" THEN T=L1:L1=3ØØ
62Ø NEXT : IF T = Ø THEN 54Ø
63Ø LN$ = LN$ + CHR$ (34) + "+" + MID$(R$,L + ~
    1,T - L) + "+" + CHR$ (34)
631 L = T: GOTO 55Ø
7ØØ IF L > LEN (R$) - 2 THEN 54Ø
71Ø T = Ø: FOR L1 = L + 2 TO LEN (R$): IF MID$
    (R$,L1,1)="#"THEN T=L1:L1 = 3ØØ
72Ø NEXT : IF T = Ø  THEN 54Ø
73Ø LN$=LN$+CHR$(34)+"+STR$(" + MID$ (R$,L + 1
    ,T - L - 1) + ")" + "+" +CHR$(34)
731 L = T: GOTO 55Ø
997 REM
998 REM *** A: INSTRUCTION ***
999 REM
1ØØØ LN$ = LN$ + "GOSUB 52ØØØ"           1Ø1Ø IF
     R$ = "" THEN RETURN
1Ø2Ø IF RIGHT$ (R$,1) = "$" THEN LN$ = LN$ + ":
    " + R$ + "=QI$": RETURN
1Ø3Ø LN$ = LN$ + ":" + R$ + "=VAL(QI$)": RETURN

1497 REM
1498 REM *** M: INSTRUCTION ***
1499 REM
15ØØ FOR L = 1 TO 25:M$(L) = "": NEXT : IF R$ =
     "" THEN 15ØØØ
151Ø T=1:FOR L=1 TO LEN(R$):IF MID$(R$,L,1)< > ~
    ","THEN M$(T)=M$(T)+MID$(R$,L,1)
1511 GOTO 153Ø
```

```
1520 T = T + 1
1530 NEXT
1540 FOR L = 1 TO T
1541 IF RIGHT$(M$(L),1)="$"THEN LN$=LN$+"Q$(" +
     STR$ (L) + ")=" + M$(L)+":"
1542 GOTO 1560
1550 LN$ = LN$ + "Q$(" + STR$ (L) + ")=" + CHR$
     (34) + M$(L) + CHR$ (34) + ":"
1560 NEXT :LN$ = LN$ + "GOSUB 53000": RETURN
1997 REM
1998 REM *** J: INSTRUCTION ***
1999 REM
2000 IF R$ = "" THEN 2100
2010 IF LEFT$ (R$,1) < > "*" THEN R$ = "*" + R$

2020 T = 0: FOR L = 1 TO NL: IF P$(L) = R$ THEN
     T = L:L = 2500
2030 NEXT : IF T = 0 THEN 15000
2040 LN$ = LN$ + "GOTO" + STR$ (T * 10): RETURN

2100 T = 0: FOR L = LN TO 1 STEP - 1: IF LEFT$ ~
    (P$(L),1) = "A" THEN T = L:L = 0
2110 NEXT : IF T = 0 THEN 15000
2120 GOTO 2040
2497 REM
2498 REM *** U: INSTRUCTION ***
2499 REM
2500 LN$ = LN$ + "QU=QU+1": IF LEFT$ (R$,1) < >
     "*" THEN R$ = "*" + R$
2510 T = 0: FOR L = 1 TO NL: IF P$(L) = R$ THEN
     T = L:L = 2500
2520 NEXT : IF T = 0 THEN 15000
2530 LN$ = LN$ + ":GOSUB" + STR$ (T * 10): RETU
     RN
2997 REM
2998 REM *** E: INSTRUCTION ***
2999 REM
3000 LN$ = LN$ + "IF QU=0 THEN END"
3010 PRINT LN * 10 + 5;
3020 IF C=1 THEN PRINT "IF QC=1 THEN";  3030 IF
     C=2 THEN PRINT "IF QC=0 THEN";
3040 IF EX$ < > "" THEN PRINT "IF";EX$;"THEN";
3050 PRINT "QU=QU-1:RETURN": RETURN
3497 REM
3498 REM *** C: INSTRUCTION ***
3499 REM
3500 LN$ = LN$ + R$: RETURN
3997 REM
3998 REM *** R: INSTRUCTION ***
3999 REM
4000 RETURN
4497 REM
4498 REM *** S: INSTRUCTION ***
4499 REM
4500 T = 0: FOR L = 1 TO LEN (R$): IF MID$ (R$,
     L,1) = "," THEN T = L:L = 255
4510 NEXT :LN$ = LN$ + "POKE 768,QS(" + LEFT$ (
     R$,T - 1) + ")
4511 POKE 769," + RIGHT$ (R$, LN (R$) - T) + ":
     CALL 770": RETURN
4997 REM
4998 REM *** G: INSTRUCTION ***
4999 REM
5000 IF R$ = "" THEN LN$ = LN$ + "POKE -16304,0
     :POKE -16297,0": RETURN
5009 REM      FIND LOOPS
5010 F = 0:IF VAL(R$) > 0 THEN LN$=LN$+ "FOR Q1
     =1 TO"+STR$( VAL (R$)) + ":":F=1
5011 R$ = LEFT$ (R$, LEN (R$) - 1)
5012 R$ = RIGHT$ (R$, LEN (R$) - LEN (STR$ ( VA
     L (R$))) - 1)
5019 REM      FIND INDIVIDUAL COMMANDS
5020 FOR L = 1 TO 6:GL$(L) = "": NEXT
5030 T = 1: FOR L = 1 TO LEN (R$)
5040 IF MID$ (R$,L,1) < > ";" THEN GL$(T) =GL$(
     T) + MID$ (R$,L,1): GOTO 5060
5050 T = T + 1
5060 NEXT
5069 REM      TRANSLATE EACH COMMAND
5070 FOR L = 1 TO T
5080 GC = 0:FOR L1=1 TO 11:IF LEFT$ (GL$(L), LE
     N (G$(L1))) < > G$(L1) THEN 5110
5090 GC = L1:L1 = 11: IF GL$(L) = G$(GC) THEN G
     L$(L) = "": GOTO 5110
5100 GL$(L) = RIGHT$ (GL$(L), LEN (GL$(L)) - LE
     N (C$(GC)))
5110 NEXT
5120 IF GC = 0 THEN 15000
5130 ON GC GOSUB 5200,5250,5300,5350,5400,5450,
     5500,5550,5600,5650,5700
5140 LN$ = LN$ + ":": NEXT : IF F = 1 THEN LN$ ~
     = LN$ + "NEXT"
5150 RETURN
5199 REM      CLEAR COMMAND
5200 LN$ = LN$ + "HGR": RETURN
5249 REM      TURNTO COMMAND
5250 LN$ = LN$ + "QA=90-" + GL$(L): RETURN
5299 REM      TURN COMMAND
5300 LN$ = LN$ + "QT=" + GL$(L) + ":GOSUB 54000
     ": RETURN
5349 REM      DRAW COMMAND
5350 LN$ = LN$ + "QL=" + GL$(L) + ":GOSUB 55000
     ": RETURN
5399 REM      PEN COMMAND
5400 T$ = GL$(L): IF T$ = "UP" THEN LN$ = LN$ +
     "QP=1": RETURN
5402 IF T$ = "DOWN" THEN LN$ = LN$ + "QP=0":RET
     URN
5405 LN$ = LN$ + "HCOLOR=": IF T$ = "ERASE" THE
     N LN$ = LN$ + "QB"
5410 IF T$ = "BLACK" THEN LN$ = LN$ + "0"
5415 IF T$ = "GREEN" THEN LN$ = LN$ + "1"
5420 IF T$ = "VIOLET" THEN LN$ = LN$ + "2"
5425 IF T$ = "WHITE" THEN LN$ = LN$ + "3"
5430 IF T$ = "BLACK2" THEN LN$ = LN$ + "4"
5435 IF T$ = "RED" THEN LN$ = LN$ + "5"
5440 IF T$ = "BLUE" THEN LN$ = LN$ + "6"
5445 IF T$ = WHITE2" THEN LN$ = LN$ + "7"
5448 RETURN
5449 REM      SCREEN COMMAND
5450 GOSUB 5400:LN$ = LN$ + ":HPLOT 0,0:CALL 62
     454:QB=" + RIGHT$(LN$,1): RETURN
5499 REM      GOTO COMMAND
5500 T = 0: FOR L1 = 1 TO LEN (GL$(L)): IF MID$
     (GL$(L),L1,1) = "," THEN T = L1
5501 L1 = 255
5510 NEXT :LN$ = LN$ + "QX=" + LEFT$ (GL$(L),T ~
     - 1) + "
5511 QY=" + RIGHT$ (GL$(L), LEN (GL$(L)) - T): ~
     RETURN
5549 REM      FULL COMMAND
5550 LN$ = LN$ + "POKE -16302,0": RETURN
5599 REM      MIX COMMAND
5600 LN$ = LN$ + "POKE-16301,0": RETURN
5649 REM      QUIT COMMAND
5650 LN$ = LN$ + "TEXT": RETURN
5699 REM      GO COMMAND
5700 LN$ = LN$ + "QP=1:QL=" + GL$(L) + ":GOSUB ~
     55000:QP=0": RETURN
5997 REM
5998 REM *** B: COMMAND ***

5999 REM
6000 LN$ = LN$ + R$: RETURN
6497 REM
6498 REM *** LABEL ***
6499 REM
6500 LN$ = LN$ + "REM" + R$: RETURN
9997 REM
9998 REM *** INITIALIZE ***
9999 REM
```

```
10000 TEXT : HOME
10010 HTAB 6: INVERSE : PRINT "                            
      "
10020 HTAB 6: PRINT " ";: HTAB 34: PRINT " "
10030 HTAB 6: PRINT " ";: HTAB 9: NORMAL : PRINT
     "TURTLE PILOT TRANSLATOR";
10031 HTAB 34: INVERSE : PRINT " "
10040 HTAB 6: PRINT " ";: HTAB 34: PRINT " "
10050 HTAB 6: PRINT " ";: HTAB 14: NORMAL : PRIN
    T "BY ALAN POOLE";
10051 HTAB 34: INVERSE : PRINT " "
10060 HTAB 6: PRINT " ";: HTAB 34: PRINT " "
10070 HTAB 6: PRINT "                            
      ": NORMAL
10080 DIM P$(2500),I$(12),G$(11),GL$(6),M$(25):D
    $ = CHR$ (4)
10090 PRINT : INPUT "WHAT IS THE NAME OF THE PRO
    GRAM? ";N$: IF N$ =""THEN 10090
10100 IF RIGHT$ (N$,2) < > ".P" THEN N$ = N$ + "
    .P"
10105 PRINT : PRINT "PLEASE WAIT..."
10110 ONERR GOTO 16000
10120 PRINT D$"VERIFY"N$: PRINT D$"OPEN"N$: PRIN
    T D$"READ"N$
10130 INPUT NL
10140 FOR L = 1 TO NL:I$ = ""
10150 GET K$: IF ASC (K$) = 13 THEN P$(L) = I$:
    GOTO 10170
10160 I$ = I$ + K$: GOTO 10150
10170 NEXT : PRINT : PRINT D$"CLOSE"
10180 FOR L = 1 TO 12: READ I$(L): NEXT
10190 FOR L = 1 TO 11: READ G$(L): NEXT
10200 ONERR GOTO 15000
10210 RETURN
10220 DATA T,A,M,J,U,E,C,R,S,G,B,*
10230 DATA CLEAR,TURNTO,TURN,DRAW,PEN,SCREEN,GOT
    O,FULL,MIX,QUIT,GO
10997 REM
10998 REM *** REMOVE SPACES FROM T$ ***
10999 REM
11000 IF T$ = "" THEN RETURN
11010 T1$ = "": FOR L = 1 TO LEN (T$)
11020 IF MID$ (T$,L,1) < > " " THEN T1$ = T1$ +
    MID$ (T$,L,1)
11030 NEXT :T$ = T1$: RETURN
14997 REM
14998 REM *** ERROR ROUTINES ***
14999 REM
15000 PRINT D$"CLOSE": PRINT "ERROR IN PILOT LIN
    E NO. ";LN; CHR$ (7)
15001 PRINT D$"RUN EDITOR"
16000 PRINT D$"CLOSE": PRINT "UNABLE TO LOAD"; C
    HR$ (7) : RUN
49996 REM
49997 REM THE FOLLOWING LINES ARE NOT PART OF TH
    E TRANSLATOR, BUT ARE
49998 REM INCLUDED IN EVERY TRANSLATED PROGRAM.
49999 REM
50000 DIM Q$(25),QS(31)
50010 HCOLOR= 3:QX = 0:QY = 0:QC = - 1:QR = 40:Q
    A = 90:QQ = 3.1415927 / 180
50020 QS(1) = 192:QS(2) = 180: QS(3) = 171:QS(4)
     = 161:QS(5) = 153:QS(6) = 144
50021 QS(7) = 136:QS(8) = 129:QS(9) = 122:QS(10)
     = 115:QS(11) = 108:QS(12) =102
50022 QS(13) = 96:QS(14) = 91:QS(15) = 86
50025 QS(16) = 81:QS(17) = 76:QS(18) = 72:QS(19)
     = 68:QS(20) = 64:QS(21) = 60

50026 QS(22) = 57:QS(23) = 54:QS(24) = 50:QS(25)
     = 47:QS(26) = 45:QS(27) = 42
50027 QS(28) = 40:QS(29) = 37:QS(30) = 35:QS(31)
     = 33
50030 POKE 770,173: POKE 771,48: POKE 772,192: P
    OKE 773,136: POKE 774,208
50031 POKE 775,5: POKE 776,206: POKE 777,1: POKE
     778,3: POKE 779,240
50032 POKE 780,9: POKE 781,202: POKE 782,208: PO
    KE 783,245: POKE 784,174
50040 POKE 785,0: POKE 786,3: POKE 787,76: POKE
     788,2: POKE 789,3: POKE 790,96
50041 POKE 791,0: POKE 792,0
50050 RETURN
51000 IF QT$ = "" THEN PRINT : RETURN
51005 QT = 0: IF RIGHT$ (QT$,1) = "&" THEN QT$ =
     LEFT$(QT$, LEN (QT$) - 1):QT=1
51010 FOR Q1 = 1 TO LEN (QT$)
51011 IF MID$ (QT$,Q1,1) = " " AND PEEK (36) > Q
    R - 9 THEN GOSUB 51100
51020 PRINT MID$ (QT$,Q1,1);: NEXT : IF QT = 0 T
    HEN PRINT
51030 RETURN
51100 QF = 0: FOR Q2 = Q1 + 1 TO Q1 + QR - PEEK
    (36) - 1
51101 IF Q2 > = LEN (QT$) THEN Q2 = 1000:QF = 1:
     GOTO 51120
51110 IF MID$ (QT$,Q2,1) = " " THEN Q2 = 1000:QF
     = 1
51120 NEXT : IF QF = :0 THEN PRINT :Q1 = Q1 + 1
51130 RETURN
52000 QI$ = ""
52010 GET QK$:QT = ASC (QK$): PRINT QK$;
52020 IF QT = 13 THEN RETURN
52030 IF QT = 8 THEN 52100
52040 IF QT = 21 THEN 52200
52050 IF QT = 24 THEN PRINT CHR$ (92): GOTO 5200
    0
52060 QI$ = QI$ + QK$: IF LEN (QI$) > 245 THEN P
    RINT CHR$ (7);
52070 IF LEN (QI$) > 250 THEN PRINT CHR$ (92): G
    OTO 52000
52080 GOTO 52010
52100 IF QI$ = "" THEN PRINT : GOTO 52010
52110 IF LEN (QI$) = 1 THEN 52000
52120 QI$ = LEFT$ (QI$, LEN (QI$) -1): GOTO 5201
    0
52200 QI$ = QI$ + CHR$ (PEEK ( PEEK (40) + PEEK
    (41) * 256 + PEEK (36)) - 128)
52210 POKE 36, PEEK (36) + 1: IF PEEK (36) > QR
    - 1 THEN PRINT
52220 GOTO 52010
53000 QM = 0:QC = 0: FOR Q1 = 1 TO 25
53001 IF LEN (QI$) < LN (Q$(Q1)) OR Q$ (Q1) = ""
     THEN 53030
53010 FOR Q2 = 1 TO LEN (QI$) - LEN (Q$(Q1)) + 1

53011 IF Q$(Q1) = MID$ (QI$,Q2, LEN (Q$(Q1))) TH
    EN QC = 1:QM = Q1:Q1=25:Q2 =300
53020 NEXT
53030 NEXT : FOR Q1 = 1 TO 25:Q$(Q1) = "": NEXT
    : RETURN
54000 QA = QA - QT: IF QA > 360 THEN QA = QA - 3
    60
54010 IF QA < 0 THEN QA = QA + 360
54020 RETURN
55000 IF QP = 1 THEN 55005
55003 HPLOT QX + 139.0005, - QY + 80.0005
55005 QX = QX + QL * COS (QA * QQ):QY = QY + QL
    * SIN (QA * QQ)
55010 IF QX < - 139 THEN QX = - 139
55020 IF QX > 140 THEN QX = 140
55030 IF QY < - 111 THEN QY = - 111
55040 IF QY > 80 THEN QY = 80
55050 IF QP = 1 THEN RETURN
55060 HPLOT TO QX + 139.0005, - QY + 80.0005: RE
    TURN
```

Program 2.

```
1 REM TURTLE PILOT EDITOR
2 REM BY ALAN POOLE
```

```
10 GOSUB 20000
17 REM
18 REM *** MAIN LOOP ***
19 REM
20 PRINT : PRINT "<";: GOSUB 100: GOSUB 12000
   : IF I$ = "" THEN 20
25 IF LEFT$ (I$,3) = "CAT" THEN PRINT D$"CATA
   LOG": GOTO 20
28 IF LEFT$ (I$,3) = "PR#" THEN PRINT D$"PR#"
   VAL ( RIGHT$ (I$,1)): GOTO 20
30 GOSUB 500: IF C = Ø THEN PRINT SE$: GOTO 2
   Ø
40 ON C GOSUB 1000,2000,3000,4000,5000,6000,7
   000,8000,9000
50 GOTO 20
97 REM
98 REM *** INPUT ***
99 REM
100 I$ = ""

110 GET K$:K = ASC (K$): PRINT K$;
120 IF K = 13 THEN RETURN : REM RETURN KEY
130 IF K = 8 THEN 200: REM BACKSPACE
140 IF K = 21 THEN 300: REM RETYPE
150 IF K = 24 THEN PRINT CHR$ (92): GOTO 100:
    REM CTRL-X
160 IF K = 27 THEN 15000: RM ESC
170 I$ = I$ + K$: IF LEN (I$) > 195 THEN PRINT
    BELL$;
180 IF LEN (I$) > 200 THEN PRINT CHR$ (92): GO
    TO 100
190 GOTO 110
197 REM
198 REM *** BACKSPACE ***
200 IF I$ = "" THEN PRINT : GOTO 110
210 IF LEN (I$) = 1 THEN 100
220 I$ = LEFT$ (I$, LEN (I$) - 1): GOTO 110
297 REM
298 REM *** RETYPE ***
299 REM
300 I$ = I$ + CHR$ ( PEEK ( PEEK (40) + PEEK (
    41) * 256 + PEEK (36)) - 128)
310 POKE 36, PEEK (36) + 1: IF PEEK (36) > 39
    THEN PRINT
320 GOTO 110
497 REM
498 REM *** DETRMINE NO. OF EDITOR COMMAND ***

499 REM
500 C = Ø: FOR L = 1 TO 9: IF C$(L) = LEFT$ (I
    $, LEN (C$(L))) THEN C = L
510 NEXT : IF C = Ø THEN RETURN
519 REM MAKE I$ THE PART RIGHT OF COMMAND
520 IF I$ = C$(C) THEN I$ = "": RETURN
530 I$ = RIGHT$ (I$, LEN (I$) - LEN (C$(C))):
    GOSUB 12000: RETURN
997 REM
998 REM *** ADD ***
999 REM
1000 IF I$ = "" THEN 1030
1010 IF VAL (I$) > LN OR VAL (I$) < 1 THEN PRIN
    T RE$: RETURN
1020 LN = VAL (I$)
1030 IF LN > 2499 THEN PRINT "TOO MANY LINES";B
    ELL$: RETURN
1040 LT = LN + 1: GOSUB 10000:LN = LT:P$(LN) =
    I$: GOTO 1030
1997 REM
1998 REM *** LIST ***
1999 REM
2000 IF LN = Ø THEN PRINT : RETURN
2010 IF LEFT$ (I$,1) = "*" THEN GOSUB 13000: GO
    TO 2050
2020 IF I$ = "" THEN FL = 1:LL = LN: GOTO 2050
2030 GOSUB 11000
2040 IF FL < 1 OR FL > LN OR LL < 1 OR LL > LN
    THEN PRINT RE$: RETURN
2050 FOR L = FL TO LL: HTAB 5 - LEN (STR$ (L)):
    PRINT L;" ";
2051 IF LEFT$ (P$(L),1) = "*" THEN 2057
2055 PRINT " ";: IF LEFT$ (P$(L),1) < > "R" THE
    N PRINT " ";
2057 PRINT P$(L)
2060 IF PEEK ( - 16384) = 141 THEN L = 2500

2070 NEXT : RETURN
2997 REM
2998 REM *** EDIT ***
2999 REM
3000 IF I$ = "" THEN PRINT SE$: RETURN
3010 GOSUB 11000
3020 IF FL < 1 OR FL > LN OR LL < 1 OR LL > LN
    THEN PRINT RE$: RETURN
3030 HOME : FOR L1 = FL TO LL: VTAB 18: HTAB 5
    - LEN ( STR$ (L1))
3031 PRINT L1;" ";P$(L1);: VTAB 18: HTAB 6: GOS
    UB 100: GOSUB 10010
3040 VTAB 24: FOR L2 = 1 TO LEN (P$(L1)) / 40 +
     2: PRINT : NEXT :P$(L1) = I$

3050 NEXT : RETURN
3997 REM
3998 REM *** INSERT ***
3999 REM
4000 LT = VAL (I$): IF LT < 1 OR LT> LN THEN PR
    INT RE$: RETURN
4010 IF LN > 2499 THEN PRINT "TOO MANY LINES";B
    ELL$: RETURN
4020 GOSUB 10000
4030 FOR L = LN + 1 TO LT STEP - 1: P$(L) = P$(
    L - 1): NEXT
4040 P$(LT) = I$: LN = LN + 1: RETURN
4997 REM
4998 REM *** DELETE ***
4999 REM
5000 IF I$ = "" THEN PRINT SE$: RETURN
5010 IF LEFT$ (I$,1) = "*" THEN GOSUB 13000: GO
    TO 5030
5020 GOSUB 11000: IF FL < 1 OR FL > LN OR LL<1
    OR LL> LN THEN PRINT RE$: RETURN
5030 FOR L = FL TO LN - (LL - FL + 1):P$(L) = P
    $(L + (LL - FL + 1)): NEXT
5031 LN = LN - (LL - FL + 1)
5040 RETURN
5997 REM
5998 REM *** NEW ***
5999 REM
6000 FOR L = 1 TO LN:P$(L) = "": NEXT :LN = Ø:
    RETURN
6997 REM
6998 REM *** LOAD ***
6999 REM
7000 N$ = I$: IF N$ = "" THEN PRINT SE$: RETURN

7010 ONERR GOTO 21000
7020 IF RIGHT$ (N$,2) < > ".P" THEN N$ = N$ + "
    .P"
7030 PRINT D$"VERIFY"N$: PRINT D$"OPEN"N$: PRIN
    T D$"READ"N$
7040 INPUT T
7050 FOR L = 1 TO T
7052 GET T$: IF T$ = CHR$ (13) THEN 7058
7055 P$(LN + L) = P$(LN + L) + T$: GOTO 7052
7058 NEXT : PRINT
7060 PRINT D$"CLOSE"
7070 POKE 216,Ø: REM RESET ONERR
7080 LN = LN + T: RETURN
7997 REM
7998 REM *** SAVE ***
7999 REM
8000 IF I$ = "" THEN PRINT SE$: RETURN
8003 ONERR GOTO 22000
8005 IF RIGHT$ (I$,2) < > ".P" THEN I$ = I$ + "
    .P"
8010 PRINT D$"OPEN"I$: PRINT D$"WRITE"I$
8030 PRINT LN
8040 FOR L = 1 TO LN: PRINT P$(L): NEXT
8050 PRINT D$"CLOSE"I$
8060 POKE 216,Ø: RETURN
```

```
8997 REM
8998 REM *** MEM ***
8999 REM
9000 PRINT "THERE ARE "; FRE (0) - 3500;" BYTES
     LEFT.": RETURN
9997 REM
9998 REM *** INPUT PROGRAM LINE ***
9999 REM
10000 HTAB 5 - LEN ( STR$ (LT)): PRINT LT;" ";:
    GOSUB 100: GOSUB 12000
10010 IF I$ = "" THEN POP : RETURN
10020 F = 0:P = 0: FOR L = 1 TO LEN (I$):T$ = MI
    D$ (I$,L,1): IF T$ = ":"THENF=1
10030 IF T$ = "(" THEN P = P + 1
10040 IF T$ = ")" THEN P = P - 1
10050 NEXT:IF (F = 0 AND LEFT$ (I$,1) < > "*")OR
     P< > 0THEN PRINT SE$:GOTO10000
10060 F = 0: FOR L = 1 TO 12: IF IN$(L) = LEFT$
    (I$,1) THEN F = 1:L = 12
10070 NEXT : IF F = 0 THEN PRINT SE$: GOTO 10000

10080 RETURN
10997 REM
10998 REM *** FIND TWO NOS. DIVIDED BY COMMA ***

10999 REM
11000 FL = VAL (I$):LL = 0: FOR L = 1 TO LEN (I$
    ) - 1
11001 IF MID$ (I$,L,1) = "," THEN I$ = RIGHT$ (I
    $, LEN (I$) - L):LL = VAL (I$)
11010 NEXT : IF LL = 0 THEN LL = FL
11997 REM
11998 REM *** REMOVE LEADING SPACES ***
11999 REM
12000 IF I$ = "" THEN RETURN
12010 IF LEFT$ (I$,1) < > " " THEN RETURN
12020 IF I$ = " " THEN I$ = "": RETURN
12030 I$ = RIGHT$ (I$, LEN (I$) - 1): GOTO 12000

12997 REM
12998 REM *** FIND FIRST AND LAST LINES OF A MOD
    ULE ***
12999 REM
13000 FOR L = 1 TO LN: IF P$(L) = I$ THEN FL = L
    : GOTO 13020
13010 NEXT : PRINT "LABEL NOT FOUND";BELL$: POP
    : RETURN
13020 LL = 0: FOR L = L TO LN: IF LEFT$ (P$(L),1
    ) = "E" THEN LL = L:L = 2500
13030 NEXT : IF LL = 0 THEN LL = LN
13040 RETURN
14997 REM
14998 REM *** END PROGRAM ***
14999 REM
15000 PRINT : PRINT "DO YOU WANT TO SAVE THE PRO
    GRAM? (Y/N)";: GET I$
15001 IF I$ = CHR$ (13) THEN PRINT : GOTO 100
15010 IF I$ < > "Y" AND I$ < > "N" THEN 15000
15015 PRINT I$
15020 IF I$ = "Y" THEN INPUT "WHAT IS THE NAME O
    F THE PROGRAM? ";I$: GOSUB 8000
15030 PRINT : PRINT "DO YOU WANT TO TRANSLATE A"
    : PRINT "PROGRAM?(Y/N)";:GETI$
15031 IF I$ < > "Y" AND $ < > "N" THEN 15030
15040 PRINT I$: IF I$ = "Y" THEN PRINT CHR$ (4)"
RUN TRANSLATR"
15050 HOME : END
19997 REM
19998 REM *** INITIALIZE ***
19999 REM
20000 TEXT : HOME
20010 HTAB 8: INVERSE : PRINT "
    "
20020 HTAB 8: PRINT " ";: HTAB 32: PRINT " "
20030 HTAB 8: PRINT " ";: HTAB 11: NORMAL : PRIN
    T "TURTLE PILOT EDITOR";
20031 HTAB 32: INVERSE : PRINT " "
20040 HTAB 8: PRINT " ";: HTAB 32: PRINT " "
20050 HTAB 8: PRINT " ";: HTAB 13: NORMAL : PRIN
    T "BY ALAN POOLE";: HTAB 32
20051 INVERSE : PRINT " "
20060 HTAB 8: PRINT " ";: HTAB 32: PRINT " "
20070 HTAB 8: PRINT "                        ":
    NORMAL
20080 DIM P$(2500),C$(9),IN$(12)
20090 BELL$ = CHR$ (7):D$ = CHR$ (4):SE$ = "SYNT
    AX ERROR" + BELL$
20091 RE$ = "LINE NO. OUT OF RANGE" + BELL$
20100 FOR L = 1 TO 9: READ C$(L): NEXT
20110 FOR L = 1 TO 12: READ IN$(L): NEXT
20120 RETURN
20130 DATA ADD,LIST,EDIT,INSERT,DEL,NEW,LOAD,SAV
    E,MEM
20140 DATA T,A,M,J,U,E,C,R,S,G,B,*
20997 REM
20998 REM *** ERROR ROUTINES ***
20999 REM
21000 POKE 216,0: PRINT D$"CLOSE": PRINT "UNABLE
    TO LOAD";BELL$: GOTO 20
22000 POKE 216,0: PRINT D$"CLOSE": PRINT "DOS ER
    ROR";BELL$: GOTO 20
```
©

TOLL FREE
Subscription
Order Line
800-334-0868
In NC 919-275-9809

A solid state disk emulator for your APPLE*, TRS80, S100, or SS50 computer**

- A FAST mass storage device. Speeds up any program requiring disk access.
- No head seek time, no motor startup time, no moving parts.
- Standard 256K bytes of storage expandable to 1 megabyte.
- Independent regulated power supply.
- Automatic power failure detect and battery backup.
- Hardware error detection and write protect.
- Only 4 bytes-ports of address I/O space used
- Hardware optimized for block transfers and access.
- Drivers, diagnostics, and utilities software provided.
- Certified to comply with Class B limits Part 15 of FCC rules.

Introductory Price $1095.
plus tax and shipping

SEND FOR FREE BROCHURE
Identify your model.

PION, INC. Tel. (617)648-1717
74 Appleton St., Arlington, MA 02174

*Trade Mark Apple **Trade Mark Tandy Corp.

There are times when you'll want to process other kinds of disk files besides Text files. The technique is illustrated with a useful cross-reference program which shows where and how often variables are used in a BASIC program.

Process Any Apple Disk File

Keith Falkner
Venice, FL

Apple's Disk Operating System recognizes four types of files: Applesoft, Integer, Binary, and Text. When the DOS command CATALOG is entered, the names of all files on the disk are displayed, and the type of each file is indicated by a letter A, I, B, or T. A-files are of course Applesoft programs and are stored by the DOS command: SAVE programname. Similarly, I-files are programs in Integer BASIC. B-files are merely copies of memory onto disk, although they are often machine-language programs or subprograms. T-files are the only genuine data files, and these have invariably been written by programs.

Apple DOS contains a very sensible restriction: a program may OPEN only a Text file. Investigation verifies that all the other types of files usually contain many null bytes, that is, bytes with no bits on, or in Applesoft, CHR$(0). Unless a program explicitly writes CHR$(0), a Text file will never contain a null byte. So when data is being read from an open file into memory, Apple DOS tests each byte transferred. If a null byte is found, Apple DOS assumes that the program has read beyond the end of data in the Text file and issues the error message END OF DATA IN #####, where ##### is a line-number.

Many Good Things

This restriction really is a great nuisance, because there are many good things we could do if only a program could process the other types of files. For instance, a program could print a program listing neatly, produce a cross-reference report, or devise some documentation from the REM statements in an Applesoft or Integer BASIC program.

In fact, a program can circumvent the restriction and OPEN any type of file, by patching DOS as follows:

POKE 42948,234: POKE 42949,169: POKE 42950,0 : REM DOS 3.2 OR 3.3 IN 48K. This changes the instruction at $A7C4 from EOR $B5C2 into NOP and LDA #0, and thus circumvents the test for type of file.

Those POKEs are effective until DOS is rebooted (via PR#6 for example). If the DOS command INIT is issued after the POKEs, the disk so initialized will contain the patch permanently, thus the version of DOS on it will never be able to issue the error message FILE TYPE MISMATCH for the OPEN command.

Figure 1. The Cross-Reference Program's Variables

```
A     200 280    330 340 420 430
A$    330 490    520 530 540 550 620 630
A$(   100 360    370 380 390
B     210 220    260 350 360 370 380 390
B$    510 530    550 630 640
B$(   100 510
C     310 400    420 440 450 460 470 480 580
C$    510 550    630 670
C(    100 140    150 160 340
C1    310 340    400 450 470
C2    150 230    310 400 410 480 610
C9    340 440    450 470 500 510
J     140 150    220 250 260 360 370 380 680
K     220 250    260 350 370 380 390 610 620 630
L     300 310
L$    220 240    310
M$    360 370    390 400 480 490
P$    180
Q     200 280    300 330 520 690
Q$    330 520    690
QQ$   120 540
S$    120 310    640 660 670
X     220 260
X$    230 240    250 260 600 630 640
X$(   100 250    260
Y     640 650    660
Z     580 650
Z$    580
```

Figure 2. The Cross-Reference Program's Line Numbers

```
210  500  520
250  250
280  210
330  500  530
390  350  370
400  360
410  340
450  410
490  470
500  420  430  440  460  480
510  500
520  540  550
560  280
630  610
650  630
670  650
690  200  280  300  330  520
700  190
```

Any program using the above technique to OPEN a non-Text file must be prepared to detect and process the expected null bytes; therefore, the commands GET and INPUT will not work. Fortunately there is a convenient routine in DOS, and here is how to use it:

PRINT CHR$(4) "READ" filename
CALL 42636 : Q=PEEK (46531)

The above line corresponds to a GET statement and delivers in Q the ASCII value of one byte. If the byte is a null byte, then Q will simply be zero, and that can be processed as easily as any other value. DOS will not issue the error message END OF DATA, unless the program reads a byte past the last sector containing the disk file.

The final tip is therefore how to detect end-of-data when processing a file other than a Text file. Actually that is the easiest part, and is simple to deal with when the file has just been opened. Both A-files and I-files start with a two-byte counter which indicates how many bytes of data are in the program, i.e., how many bytes to process. B-files contain this same counter, but before it is another two-byte counter which tells where the image of memory is to be loaded. As usual with 6502 software, all these two-byte counters have the less significant byte first and the more significant last.

Illustrating these techniques is an adaptation of a program which first appeared in **COMPUTE!**, May/June 1980. The program prints a cross-reference list of either the variables or line-numbers in an Applesoft program. The program is very handy and was admirably explained by its esteemed author.

```
100 TEXT : HOME : NORMAL : DIM A$(15),B$(3),X$
    (500),C(255)
110 PRINT "CROSS-REF    JIM BUTTERFIELD": PRIN
    T
120 QQ$ = CHR$ (34):S$ = "      ":B$(1) = Q$:B$
    (3) = CHR$ (58)
130 INPUT "VARIABLES OR LINES? ";Z$:C2 = 5: IF
     ASC (Z$) = 76 THEN C2 =6
140 FOR J = 1 TO 255:C(J) = 4: NEXT : FOR J =
    48 TO 57:C(J) = 6: NEXT
150 IFC2=5THENFORJ=65TO90:C(J)=5:NEXT:C(36)=7:
    C(37)=7:C(40)=8
160 C(34) = 1:C(178) = 2:C(131) = 3
170 PRINT : INPUT "PROGRAM NAME: ";P$
180 IF P$ < "A" THEN PRINT CHR$ (4)"CATALOG":
    GOTO 170
190 GOSUB 700: PRINT CHR$ (4)"OPEN"P$: PRINT C
    HR$ (4)"READ"P$
200 GOSUB 690:A = Q: GOSUB 690: PRINT 256 * Q
    + A" BYTES"
210 IF B = 0 GOTO 280
220 PRINT L$;:K = X: FOR J = B TO 1 STEP - 1:
    PRINT " ";A$(J);:X$ = A$(J)
230 IF C2 = 6 AND LEN (X$) < 5 THEN X$  = " " 
    + X$: GOTO 230
240 X$ = X$ + L$
250 IF X$(K) > = X$ THEN X$(K + J) = X$(K):K =
    K - 1: GOTO 250
260 X$(K + J) = X$: NEXT J:X = X + B: PRINT :B
    = 0
270 REM : GET NEXT LINE, TEST END
280 GOSUB 690:A = Q: GOSUB 690: IF A + Q = 0 G
    OTO 560
290 REM GET LINE NUMBER
300 GOSUB 690:L = Q: GOSUB 690:L = Q * 256 + L

310 C = C2:C1 = - 1:L$ = RIGHT$ (S$ + STR$ (L)
    ,6)
320 REM GET BASIC STUFF
330 GOSUB 690:A = Q:A$ = Q$
340 C9 = C(A): IF C9 > C1 GOTO 410
350 K = 0: IF B = 0 GOTO 390
360 FOR J = 1 TO B: IF A$(J) = M$ GOTO 400
370 IF A$(J) < M$ THEN NEXT J:K = B: GOTO 390
380 FOR K = B TO J STEP - 1:A$(K + 1) = A$(K):
     NEXT K
390 B = B + 1:A$(K + 1) = M$
400 C = C2:C1 = - 1:M$ = ""
410 IF C2 = 5 GOTO 450
420 IF A = 171 OR A = 172 OR A = 176 OR A = 19
    6 THEN C = 6: GOTO 500
430 IF A = 44 OR A = 32 GOTO 500
440 IF C9 < > 6 THEN C = 9: GOTO 500
450 IF C9 = C THEN C = - 1:C1 = 4
460 IF C > 6 GOTO 500
470 IF C < 0 AND C9 > C1 AND C9 > 6 THEN C1 =
    C9: GOTO 490
480 IF C2 = 5 THEN IF LEN (M$) > 2 OR C > 0 GO
    TO 500
490 M$ = M$ + A$
500 ON C9 + 1 GOTO 210,510,510,510: GOTO 330
510 B$ = B$(C9):C$ = ""
520 GOSUB 690:A$ = Q$: IF Q = 0 GOTO 210
530 IF A$ = B$ GOTO 330
540 IF A$ < > QQ$ GOTO 520
550 A$ = B$:B$ = C$:C$ = A$: GOTO 520
560 PRINT : PRINT CHR$ (4)"CLOSE"
570 INPUT "PRINTER? ";Z$
580 C= 3:Z = 6: IF ASC (Z$) = 89 THEN C = 4:Z
    = 12: PRINT CHR$ (4)"PR#1"
590 PRINT : PRINT "CROSS REFERENCE - PROGRAM "
    ;P$
600 X$ = "": FOR J = 1 TO X:A$ = X$(J)
610 IF C2 = 6 THEN K = 6: GOTO 630
620 FOR K = 1 TO LEN (A$): IF MID$ (A$,K,1) <
    > " " THEN NEXT K: STOP
630 B$ = LEFT$ (A$,K - 1):C$ = MID$ (A$,K + 1)
    : IF X$ = B$ GOTO 650
640 PRINT :Y = 0:X$ = B$: PRINT X$; MID$ (S$,1
    ,5 - LEN (X$));
650 Y = Y + 1: IF Y < Z GOTO 670
660 Y = 1: PRINT : PRINT S$;
670 PRINT LEFT$ (S$,6 - LEN (C$));C$;
680 NEXT J: PRINT : PRINT CHR$ (4)"PR#0"; END
690 CALL 42636:Q = PEEK (46531):Q$ = CHR$ (Q):
     RETURN
700 POKE 42948,234: POKE 42949,169: POKE 42950
    ,0: RETURN
```

©

TOLL FREE
Subscription
Order Line
800-334-0868
In NC 919-275-9809

Whole Brain Spelling For The Apple II

Whole Brain Spelling is the first in a series of educational software packages from SubLOGIC. The program has been designed to help the user develop internal visualization skills for improving spelling in a manner as entertaining as it is educationally sound. It effectively utilizes the graphic and color capabilities of the Apple II Plus computer to provide positive user feedback and to emphasize visual aspects of the learning process.

The program itself is extremely user friendly. You can move to any lesson section as desired, choose your own word lists to study, and proceed at your own rate. A main Spelling Menu is accessible from any portion of the program. Lesson instructions are also always available at the touch of a key.

A 2000-word list of practice spelling words is included with Whole Brain Spelling, organized in order of increasing spelling difficulty. Study words can be printed in upper- or lowercase, in any color you choose. And each correct spelling rewards you with a rainbow of varying, multicolored letters until the next word is selected.

The standard Whole Brain Spelling program comes complete with 200 ten-word lists, systematically chosen and organized in order of increasing spelling difficulty. Whole Brain Spelling is also available with supplementary word lists in the following categories: Medical, Scientific, Secretarial, Fairy Tale, and A Child's Garden of Words.

The Medical word lists are helpful to anyone involved in the health sciences or the medical profession, and include categories in general medical terms, diagnosis, anatomy, and drugs.

The Scientific word lists are divided into the categories of general science, earth science, life science, and physical science. Each category is graded into four levels of difficulty.

The Secretarial word lists include commonly used real estate, insurance, legal, commercial, and accounting terms organized into four difficulty levels.

For anyone who is a child at heart, the list of Fairy Tale words should prove irresistible. Taken from Grimm's and other fantasy tales, these words will pique your curiosity as well as enrich your spelling skills.

A Child's Garden of Words gives the preschooler through third grader (ages 5-9) a head start in the development of word visualization skills. These are the words most often encountered in beginning reading lessons, and these lists offer a primer to the main word lists found on the standard Whole Brain Spelling program.

A2-ED1 Whole Brain Spelling is available at most computer stores or from SubLOGIC direct. The A2-ED1 requires 48K memory and either an Apple II Plus or an Apple II with Applesoft in ROM (a color monitor is also recommended). Unless you request otherwise, the main word list will be included with each program ordered. The disk price is $34.95. For direct orders add $1.50 for shipping and specify UPS or first class mail. Illinois residents add 5% sales tax.

SubLOGIC Communications Corp.
713 Edgebrook Drive
Champaign, IL 61820
(217)359-8482

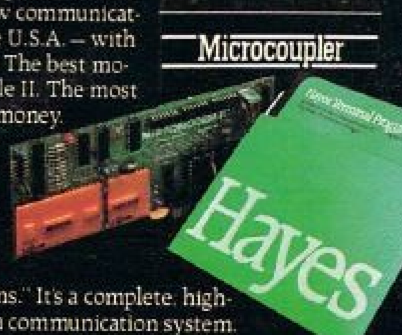
Scholastic Aptitude Test Package For Atari

Program Design has released *Preparing for the SAT* for Atari 400 and 800 computers. The package teaches students how to take the Scholastic Aptitude Test (SAT) and other aptitude and intelligence tests, develops problem-solving skills, and provides practice in answering questions typically found on such tests.

Preparing for the SAT consists of six cassettes, a user's manual, and a copy of the booklet "Making the Grade," all packaged in a special storage container. The six cassettes are:

- "Taking Aptitude Tests" – This cassette explains the purpose of standardized IQ and aptitude tests. It discusses some of the false beliefs surrounding such tests. It describes ways to improve test-taking skills. And it presents a strategy for answering those questions that are most likely to pay off with correct answers.
- "Vocabulary Builders I and II" – These two cassettes are designed to help develop a student's vocabulary and to build skills needed to answer synonym and antonym questions.

More Apple II owners choose Hayes Micromodem II than any other modem in the world. Compare these features before you buy. You should. It's your money. Thousands of other Apple II owners have already compared, considered, and are now communicating – all over the U.S.A. – with Micromodem II. The best modem for the Apple II. The most modem for your money.



A complete data communication system. Micromodem II is not "base priced" plus necessary "options." It's a complete, high-performance data communication system. The printed circuit board fits – quickly and easily – into your Apple II, eliminating the need for a serial interface card. And the Microcoupler™ (included) connects the Apple II directly to a standard modular telephone jack. Auto-dial and -answer features are built-in. Operation can be full or half duplex, with a transmission rate of 300 bps. And it's Bell 103 compatible and FCC approved.

Now there's Hayes Terminal Program, too! Developed by Hayes specifically for Micromodem II, this new Terminal Program allows you to access all the great features of your modem in a matter of seconds.

With it, you can use your CP/M®, DOS 3.3 or Pascal formatted diskettes to create, send, receive, list and delete files. Hayes Terminal Program is a complete, stand-alone disk.

And because it's menu driven, you can choose from a wide variety of options to set your communication parameters – as well as change hardware configuration – directly from the keyboard. It even allows you to generate ASCII characters that are normally not available from Apple keyboards, further extending your capabilities. Incoming data can be printed (on serial or parallel printers) as it's displayed on your screen.

Software sold with Micromodem II or separately. A Terminal Program disk and user manual now come with Micromodem II; or, if you already have one, you can buy the Terminal Program separately.

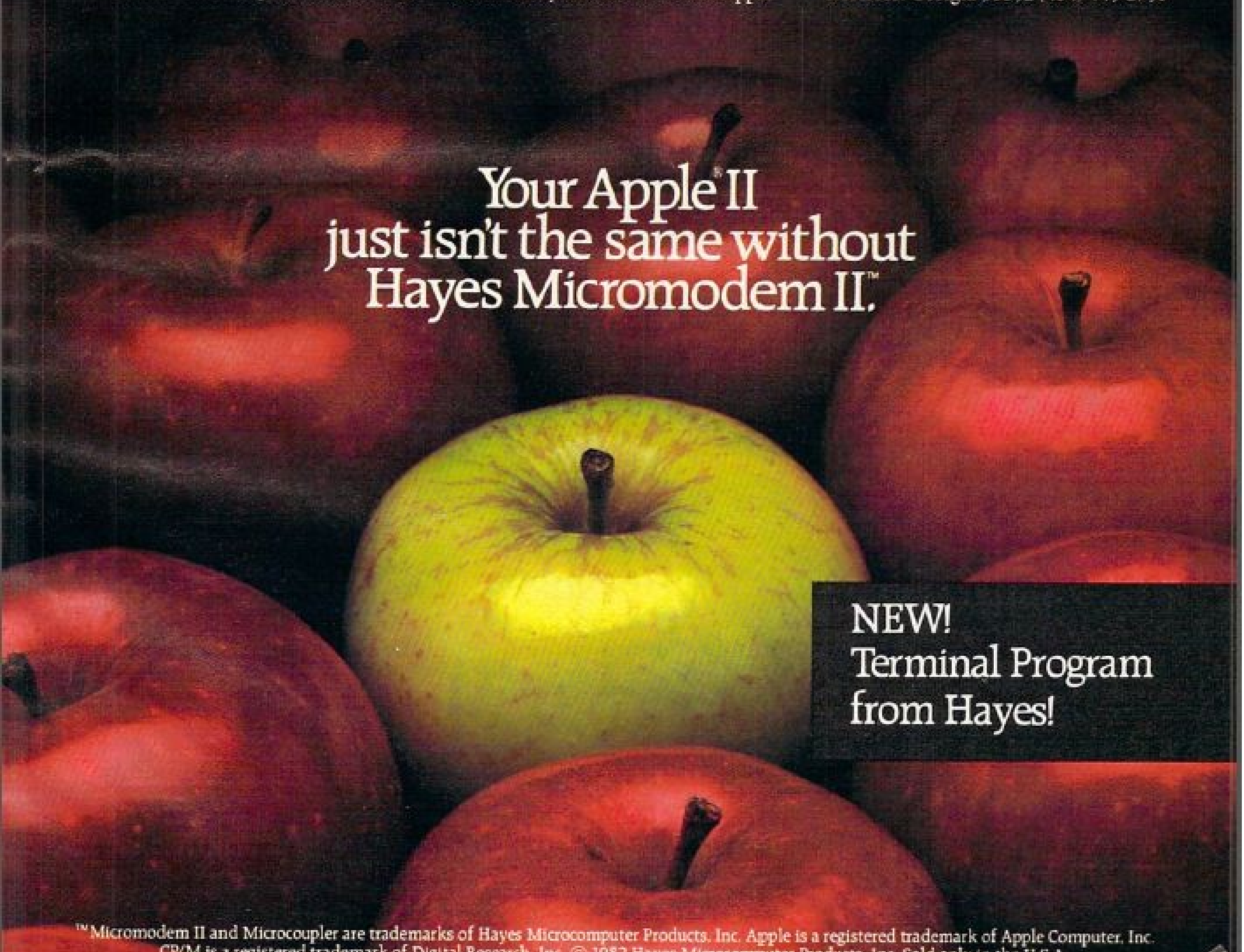
If you're ready to communicate with other computers, to access information utilities, timesharing systems, or use bulletin boards, then you're ready for Micromodem II. Come on. Compare. Consider.

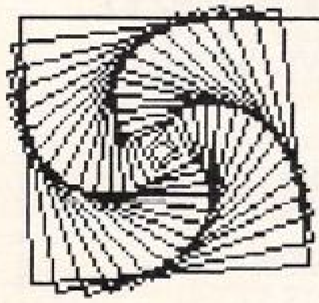
Micromodem II is already the best-selling modem for the Apple II. Now, with Hayes' new Terminal Program, it's better than ever.

Hayes

Don't settle for anything less. Available at computer stores all across America.

Hayes Microcomputer Products, Inc.
5835 Peachtree Corners East
Norcross, Georgia 30092 (404) 449-8791





A Monthly Column

Friends Of The Turtle

David D. Thornburg
Associate Editor

LOGO Is Not Just Child's Play

Revolutionary periods are more than a time of change – they are often a time of great confusion as well. Those of us who are excited about the emergence of computer languages tailored to people's needs may be less sensitive than we should be to the way our message is being received.

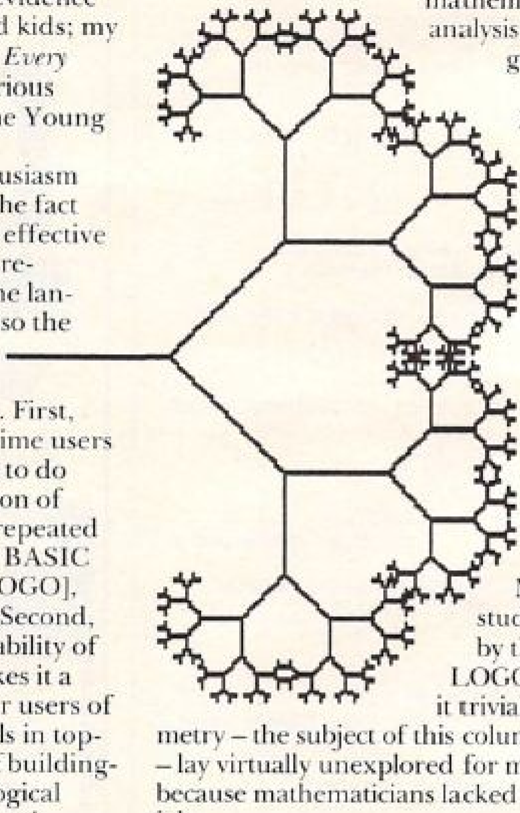
In the case of LOGO and Atari PILOT, this has had unfortunate consequences. Several readers have written to suggest that LOGO and Atari PILOT are "kid's" languages and are thus not worthy of serious attention. They cite as evidence Papert's *Mindstorms*, a book on LOGO and kids; my books, *Picture This!*, *Picture This Too!*, and *Every Kid's First Book of Robots and Computers;* various magazine articles; the very existence of the Young People's LOGO Association, etc.

Admittedly, much of the public enthusiasm for these languages has been devoted to the fact that, like English, PILOT and LOGO are effective communication tools for children. Let us remember, however, that while English is the language for "Baa, Baa, Black Sheep," it is also the language for James Joyce's *Ulysses* – the latter is definitely not for children.

The key to LOGO's power is twofold. First, much of it is very easy to learn, and first-time users find that within a short time they are able to do "interesting" things. (To me, the generation of logarithmic spirals is interesting, but the repeated printing of my name on the screen is not. BASIC has an easy time with the latter [as does LOGO], and has a horrible time with the former.) Second, LOGO is extensible by the user. This capability of LOGO, while of utility to youngsters, makes it a tremendous problem-solving language for users of any age. LOGO users readily develop skills in top-down programming and in the creation of building-block procedures that not only impart a logical order to programs, but also make them much easier to debug. When one adds to this such features as recursion, local variables, and list manipulation, it is obvious that LOGO is far more than just a kid's language. In fact, it is a far more useful language for many applications than many of the popular computer languages in use today.

Keep in mind that LOGO was a product of the artificial intelligence community. I assure you that something that is just a kid's language does not hold the interest of the MIT computer science department for over a decade.

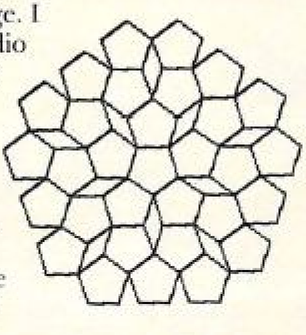
Is turtle geometry easy to use? Of course it is. But, do LOGO's detractors know that finite differential geometry (turtle geometry's formal name) is a major tool for exploring some aspects of pure mathematics that have evaded analysis by traditional analytic geometry?



Those who think that LOGO is only for kids should read *Turtle Geometry* by Abelson and diSessa. If their treatment of relativity theory is too tame for your kids, try reading Buckminster Fuller's *Synergetics* (Fuller independently developed finite differential geometry and used it to make some very interesting discoveries). My Stanford graduate students get slowed down by that book, but perhaps LOGO's detractors will find it trivial reading. Fractal geometry – the subject of this column a few months back – lay virtually unexplored for more than 50 years because mathematicians lacked the tools to do the job.

This July it was my pleasure to give a lecture on the consequences of dimensionality on the conservation rules of geometry. Apple LOGO was my principal tool.

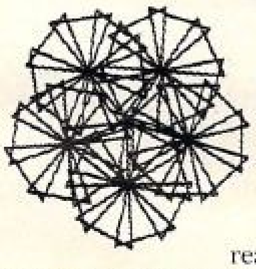
The reason I even care about this argument is that it has the promise of becoming a self-fulfilling prophecy. I have heard Apple dealers tell customers that LOGO is a kid's language. I have seen languages like Radio Shack Color LOGO that are excellent turtle graphics environments, but lack the list manipulation and other features that characterize a full LOGO implementation. In short, I have seen much confusion in the marketplace regarding LOGO and Atari PILOT.



So, please, know that languages like LOGO are marvelous tools for children – and that they are marvelous tools for almost everyone else as well. The power of a good tool is restricted only by the capabilities of its user. LOGO is a good tool.

Those Logophobes who feel like giving the language a second chance should read Harold Abelson's new book from Byte/McGraw Hill. The book is published in two editions, *Apple Logo* (for the LCSI LOGO sold by Apple), and *Logo for the Apple II* (for the MIT LOGO sold by Terrapin and Krell).

This book is excellent for all LOGO users simply because it is far more than a reference work. Abelson has managed to combine descriptions of LOGO primitives with projects that deepen the user's familiarity with the language. The first 60 pages are devoted to turtle graphics, and the remaining 150 concentrate on the other aspects of LOGO that make it a complete computer language. Thus, in addition to turtle graphics, readers become well versed in list manipulation, recursion, hierarchical structures, etc. While it is fair to say that no prior experience in programming is required to read this book, those of you who are learning LOGO as a replacement of or supplement to another language will not find Abelson's book excessively wordy. The text follows several presentation styles: reference material, sample procedures, and projects for the user to solve on his or her own. Except for elementary grade school children, I can't think of any LOGO users who would not benefit from this book.



How To Grapple With A Turtle

In my last column I showed Apple LOGO users how to print screen images on the Silentype printer. The Silentype has many features (low cost, quiet operation, etc.), but it doesn't produce pictures with very high contrast. For high contrast one must consider using a dot matrix impact printer.

Because I need high quality screen images for various reasons, I invested in the Grappler printer interface card (from Orange Micro) for use with my Epson MX-100 printer. I have not done an exhaustive search of the printer interfaces for the Apple II, but I can't think of much I would want to do that can't be done with the Grappler.

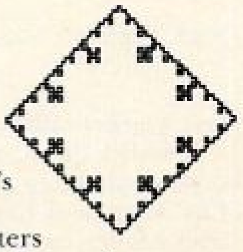
For example, this printer interface allows you to print a screen image at double size (rotated by 90 degrees) so it fits perfectly on an 8.5 by 11 sheet of paper. I enjoy the results of this print mode so much that I haven't explored any of the others.

To generate such prints for the Epson printer (there are Grapplers for other graphics printers as well), you should enter

```
SETSCRUNCH 0.84
```

before drawing any pictures. This compensates for the dot aspect ratio of the Epson printer. If you are using another printer (or another printing mode), you may have to experiment by drawing squares with various settings of SETSCRUNCH (or .ASPECT for those of you with MIT LOGO) until you get a picture that is perfectly square. The following procedure is all that is needed to generate a full-page image of your graphics screen. This procedure is written in Apple LOGO and assumes that the Grappler card is plugged in slot 1 of the Apple:

```
TO PRINTPICT
MAKE "CTRI CHAR 9
.PRINTER 1
PRINT WORD :CRI "GDR
PRINT CHAR 12
.PRINTER 0
END
```



This procedure gets the printer's attention with the character ctrl-I (CHAR 9), followed by letters that set the various options. G indicates that we want a graphics image, D means it should be double size, and R means it should be rotated by 90 degrees. If you want to use the enhanced print mode of the Epson printer, add an E to the list, and you will get a much denser print (with a longer print time, of course).

That's all there is to it! The accompanying figures are taken from my next book, tentatively titled *Discoveries of Beauty*. (This book should appear from Addison-Wesley about January 1983.) Most of the illustrations for this book were generated with the procedure shown above. As you can see, the Grappler lets your Epson printer do a fine job printing pictures generated with LOGO on your Apple computer. ©

Review:

High Orbit For Apple

Erann Gat
Oak Ridge, TN

I opened the package with anticipation. *High Orbit* seemed pretty ordinary for a computer game: a disk, some P.R. from Gebelli Software, the company that sells *High Orbit,* and a sheet of rather cryptic instructions.

I booted the disk in the usual way, and *High Orbit* immediately became very unordinary. My mouth fell open as I listened to the fastest disk boot I had ever heard. I later timed the furious "clickclickclick" of the head stepper motor: it was reading seven tracks per second! Apple DOS generally reads a track and a half per second.

The program then went into a nice demo mode which included some animated three-dimensional graphics, but nothing to give a clue as to what the game was all about. I tried for five minutes to start the game. I tried every key, but nothing worked. Oh well, when all else fails, read the directions. Aha! Control-R starts the game.

High Orbit starts with three dots that zoom onto the screen from the depths of space, which is gratifyingly free of stars. The object of the game is to "construct a space station" by moving a little fuzz ball (which represents a piece of the station) onto each of the dots using a tractor beam. To make it a bit more challenging, the dots spin around each other in a circle, and you can use the tractor beam for only a limited amount of time before it has to recharge. On top of that, there are the ubiquitous enemy spaceships that zip onto the screen and destroy your fuzz balls, so you have to start all over again. (You can destroy enemy spaceships, *if* you are fast enough, and that is a big "if.")

When (and if) you manage to maneuver a fuzz ball onto each of the dots at the same time, the space station is suddenly transformed into an abstract, three-dimensional shape which undergoes some breathtaking gyrations, splits in two, and starts spinning again.

The next phase is to "energize" the space station by moving yet another fuzz ball into the center (and I do mean the *exact* center) of the station and zapping it with your laser. Enemy spaceships will again try to destroy your supply of fuzz balls before you can get one fuzz ball into the center and destroy it.

If you are successful, the station stops spinning, becomes rainbow colored, and turns itself inside out, depositing the "crew" in deep space. The crew of the space station is just three little humanoid figures which pop onto the screen and do not move. The space station drops back into the depths of space, giving the impression that the crew is being launched into high orbit (hence the name of the program).

The last and final phase consists mainly of watching a shuttle pick up the crew. According to the instructions, you have to move the crew in front of the shuttle with your tractor beam, but I never had to. The shuttle seems to know where to go, and it will even destroy enemy spaceships that stray too close.

So how do you lose? Enemy spaceships cannot destroy you; in fact, you cannot be destroyed at all. Aye, but here's the rub: the space station must be constructed and energized before time expires. You get about two minutes to finish. If you do not, the game stops, and "mission incomplete" flashes on the screen.

If you do manage to complete a station within the time limit, you get a new station to build, but this one has four points instead of three. This goes on until you complete a six-point station. Then you go back to three points, but enemy spaceships get more aggressive. Every time you complete a station, a little colored square appears in a long hollow bar at the bottom of the screen. The bar is very long; I managed to fill up only about one-fifth of it with colored squares. You can always restart the game at the point where you last ran out of time.

High Orbit is a unique and challenging game. The graphics are well done and use the Apple's color capabilities to their fullest. It is a joy to play, provided you use a joystick. Paddles can be frustrating, and keyboard control was a frightening experience. (One nice feature of the keyboard control, though, is that you can redefine which key controls which function.) There seem to be enough levels of difficulty to keep even the best player occupied for a long, long time (although I was not able to get past the first few levels!).

All in all, *High Orbit* is an excellent game for all ages. It is challenging but not frustrating, simple but not boring. It requires a 48K Apple II with a disk drive. A joystick is not necessary, but it is *very* desirable.

High Orbit
Gebelli Software
1787 Tribute Road
Suite G
Sacramento, CA 95815
Requires 48K, disk
$29.95

©

A Word-Based Voice Synthesizer For The Apple II

David Barron
Spring Valley, NY

Since I purchased my computer I have been interested in voice synthesis. Its applications in CAI, games, and error handling seemed extensive. I decided to apply my newly learned machine language skills to writing my own voice routines.

My routines would have to meet several requirements:

1. They would have to be word based. This would keep the amount of memory per word constant. It would also provide for block memory organization. As well as this, it would simplify the program itself.
2. The routines would have to be easy to use. They would be activated by a POKE and a call, or by similar means. This would enable beginners to use the programs with ease.
3. To eliminate any excess costs, the routines would be hardware independent. They would make use of the Apple's cassette port and built-in speaker.

Memory Organization

The memory used to store a vocabulary is divided into 2000-byte blocks. Each of these blocks will be used to store eight, distinct words. Each word will be stored in its own bit of the block of memory. In other words, bit 0 stores word 0, bit 1 stores word 1, and so on. I chose to store the words this way rather than sequentially to reduce the complexity of the program. If I chose the latter way, many rotate commands would be required. These tend to get confusing, and, if you are not careful, very sloppy.

Since a single word rarely contains periods of silence, no data compression is necessary. Again, this simplifies the program. In order to store data in the correct bit, a few things must be done:

1. Load in the old byte.
2. Get a bit from the input port.
3. Move the input bit to the right position.
4. Plug this bit into the old byte.
5. Store the old byte.

Exactly how this is done will be explained in further detail later on.

How Speech Enters And Exits

The data enters into the program through the cassette and exits through the Apple on-board speaker. First let's talk about recording. Location $C060 is the *cassette in*. When a byte is read from this location, the seventh bit is affected according to the audio signal present. After this location has been sampled, the seventh bit is isolated. It is then plugged into the correct position as explained above.

When in the playback mode, your voice is produced by the on-board speaker. Because the case resonates at certain frequencies, I would recommend hooking up an external speaker, as I have. This greatly improves the quality of any sound produced by the computer, especially voice. One note: when wiring up the speaker, use shielded cable. If you do not, a tremendous amount of RF interference will occur.

The speaker is controlled by location $C030. Every time this memory location is accessed, a click is produced by the speaker. Be careful here. If you use a store instruction to toggle the speaker, it will be toggled twice. This is so because the 6502 does a read before any write. This accesses the location twice, thus producing two clicks.

Getting back to the program – once the correct data byte is loaded, the correct bit is isolated. If this bit is different than the last sample obtained, a change in state has occurred. This will result in the toggling of the speaker, producing a sound. Doing this at the proper rate reproduces the recorded word.

Here's a brief explanation of the machine language "record" and "playback" routines:

Record

The Record routine is probably the most complex part of this program. The entry point is $9000. Here is how it works:

1. All pointers are set. This includes the calculation of the position of the word and the bit that the word is located in.
2. The Y register is set to zero. This will be

the index of the indirect address of the word.

3. A delay loop is executed. This is the start of the main program loop. The delay determines the sampling rate.

4. The sample byte is taken from the cassette port. The seventh bit is then isolated via an AND instruction.

5. The X register is set to $FF if the input bit was high, or $00 if the bit was low.

6. This result is moved to the accumulator. There it is ANDed with the byte that contains the bit that the word is to be stored in high. This provides us with a byte that has the bit we want the word in set according to the cassette input. All other bits in the byte are zero. This value is saved.

7. The accumulator is loaded with the mask byte and then inverted. This forms a byte with all bits set, except for the bit that the word will be stored in.

8. The current byte is loaded and then ANDed with the previously obtained value. This leaves the byte undisturbed except for the bit that the word will be stored in. This is set to zero.

9. This value is logically ORed with the byte that contained the data sample in the proper place.

10. At this point we have successfully plugged the input sample into the current byte.

11. The current byte is now stored. We are almost finished.

12. The Y register is incremented. If it is zero, then a page has been completed. In this case the page is incremented.

13. If the last page has been done, the routine ends. If not, then it jumps back to the delay routine and goes one more time.

Play

The playback routine is far simpler than the recording routine. Its entry point is $9049.

1. All pointers are set. The positions of the word and of its bit are also calculated.

2. This is the beginning of the main loop. A delay is executed. This determines the sampling rate.

3. The Y register is zeroed. It will be the index to the indirect address.

4. The current data byte is sampled.

5. This value is ANDed with the mask byte. This results in all bits being zero except for the bit containing the word data, which is unaffected.

6. This is compared to the last data bit obtained.

7. If the value is the same, then nothing happens.

8. If there is a difference, the speaker is toggled.

9. The Y register is incremented, and the program checks whether a page has been completed.

10. If a page has been completed, the current page is incremented.

11. If the last page was done, the program ends.

12. Otherwise the program loops back until done.

Entering The Program Into Memory

Type in the BASIC Loader (Program 1) and RUN it to put the machine language program into memory. Then type CALL-151 to enter the monitor. Once this has been done, SAVE the program by typing: BSAVE VOC I.1OBJ0, A$9000,L$C3.

The next step is to generate the table used by the mask subroutine. To do this, type the following:

```
*310:01 02 04 08 10 20 40 80
```

To save it, type:

```
BSAVE TABLE,a$310,L$10
```

Using The Program

To use the program requires only three simple steps:

1. POKE 0 with the word number.
2. POKE 772 with the speed.
3. Call the appropriate routine.

A sample program would look something like this:

```
10 POKE 0,1: REM WORD
20 POKE 772,10 : REM SPEED
30 CALL 9*4096+64+9 : REM PLAY
40 END : REM DONE
```

I have included three sample programs:

Program 2: This is a simple routine that speaks any number put in. You must enter the vocabulary from Table 1 before using it.

Program 3: This is a CAI demo. It is an addition quiz that uses Program 1 as a subroutine. This program shares a vocabulary with Program 1.

Program 4: This is a vocabulary builder. It should be used to build the vocabulary in Table 1.

I hope you enjoy using these routines, as I have. They make your programs many times more pleasant and impressive.

Table 1.

WORD NUMBER	WORD	WORD NUMBER	WORD
0	ZERO	27	NINETY
1	ONE	28	HUNDRED
2	TWO	29	THAT
3	THREE	30	IS
4	FOUR	31	CORRECT
5	FIVE	32	WRONG
6	SIX	33	TRY
7	SEVEN	34	AGAIN
8	EIGHT	35	WHAT
9	NINE	36	PLUS
10	TEN	37	MINUS
11	ELEVEN	38	NEGATIVE
12	TWELVE	39	WELCOME
13	THIRTEEN	40	MATH
14	FOURTEEN	41	QUIZ
15	FIFTEEN	42	PROBLEM
16	SIXTEEN	43	NUMBER
17	SEVENTEEN	44	YOU
18	EIGHTEEN	45	GOT
19	NINETEEN	46	OUT
20	TWENTY	47	OF
21	THIRTY	48	PROBLEMS
22	FORTY	49	OR
23	FIFTY	50	PERCENT
24	SIXTY	51	HOW
25	SEVENTY	52	MANY
26	EIGHTY		

Program 1.

```
10 FOR ADRES=36864TO37055:READ DATTA:POKE ADR
   ES,DATTA:NEXT ADRES
36864 DATA 32, 153, 144, 32, 121, 144
36870 DATA 160, 0, 32, 136, 144, 173
36876 DATA 96, 192, 41, 128, 141, 5
36882 DATA 3, 162, 0, 201, 0, 240
36888 DATA 2, 162, 255, 138, 45, 0
36894 DATA 3, 141, 6, 3, 173, 0
36900 DATA 3, 73, 255, 141, 5, 3
36906 DATA 177, 1, 45, 5, 3, 13
36912 DATA 6, 3, 145, 1, 200, 192
36918 DATA 0, 208, 207, 32, 148, 144
36924 DATA 205, 2, 3, 240, 5, 160
36930 DATA 0, 76, 8, 144, 76, 191
36936 DATA 144, 32, 153, 144, 32, 121
36942 DATA 144, 160, 0, 32, 136, 144
36948 DATA 177, 1, 45, 0, 3, 205
36954 DATA 3, 3, 240, 6, 141, 3
36960 DATA 3, 174, 48, 192, 141, 3
36966 DATA 3, 200, 192, 0, 208, 229
36972 DATA 32, 148, 144, 205, 2, 3
36978 DATA 240, 75, 160, 0, 76, 81
36984 DATA 144, 169, 0, 133, 1, 173
36990 DATA 1, 3, 133, 2, 105, 8
36996 DATA 141, 2, 3, 96, 173, 4
37002 DATA 3, 141, 5, 3, 206, 5
37008 DATA 3, 208, 251, 96, 230, 2
37014 DATA 165, 2, 96, 165, 0, 41
37020 DATA 7, 170, 189, 16, 3, 141
37026 DATA 0, 3, 165, 0, 41, 24
37032 DATA 42, 105, 80, 141, 1, 3
37038 DATA 165, 0, 41, 32, 201, 0
37044 DATA 240, 8, 173, 1, 3, 105
37050 DATA 8, 141, 1, 3, 96, 96
```

Program 2.

```
10 HIMEM: 8192
15 IF  PEEK (768 + 17) = 2 THEN 50
20 PRINT  CHR$ (4);"BLOAD TABLE"
30 PRINT  CHR$ (4);"BLOAD VOC 1.1.OBJ0"
40 PRINT  CHR$ (4);"BLOAD NUMBERS.VOCAB,"
50 HOME
60 INPUT "TYPE IN YOUR NUMBER (<1000) ";N
70 GOSUB 100
80 GOTO 50
100 REM
110 IF N > 1000 OR N < >  INT (N) THEN  RETURN

130 IF N = 0 THEN RETURN
135 IF N < 21 THEN W = N: GOSUB 500: RETURN
140 IF N > 99 THEN 300
150 A1 = INT (N / 10)
160 W = A1 + 18: GOSUB 500
170 N = N - 10 * A1
180 GOTO 130
300 A1 = INT (N / 100)
310 W = A1: GOSUB 500
315 W = 28: GOSUB 500
320 N = N - A1 * 100
330 GOTO 130
500 POKE 772,17
510 POKE 0,W
520 CALL 9 * 4096 + 4 * 16 + 9: REM $9049
530 RETURN
```

Program 3.

```
10 HIMEM: 8192
15 IF PEEK (768 + 17) = 2 THEN 50
20 PRINT CHR$ (4);"BLOAD TABLE"
30 PRINT CHR$ (4);"BLOAD VOC 1.1.OBJ0"
40 PRINT CHR$ (4);"BLOAD NUMBERS.VOCAB,"
50 HOME
52 NR = 0
55 GOSUB 1000
60 GOTO 600
99 HOME
100 REM
110 IF N > 1000 OR N < > INT (N) THEN 100
130 IF N = 0 THEN RETURN
135 IF N < 21 THEN W = N: GOSUB 500: RETURN
140 IF N > 99 THEN 300
150 A1 = INT (N / 10)
160 W = A1 + 18: GOSUB 500
170 N = N - 10 * A1
180 GOTO 130
300 A1 = INT (N / 100)
310 W = A1: GOSUB 500
315 W = 28: GOSUB 500
320 N = N - A1 * 100
330 GOTO 130
500 POKE 772,17
510 POKE 0,W
520 CALL 9 * 4096 + 4 * 16 + 9: REM $9049
530 RETURN
600 FOR C = 1 TO P
605 A = INT ( RND (1) * 500):B = INT ( RND (1)
     * 500)
610 W = 35: GOSUB 500
615 PRINT "WHAT ";
620 W = 30: GOSUB 500
625 PRINT "IS ";
630 N = A: GOSUB 100
635 PRINT A;" + ";B
637 PRINT
638 FOR D = 1 TO 200: NEXT D
640 W = 36: GOSUB 500
650 N = B: GOSUB 100
```

```
660 INPUT N
662 IF N = A + B THEN NR = NR + 1: GOTO 800
665 Q = Q + 1: IF Q > 2 THEN Q = 0: GOTO 850
680 W = 33: GOSUB 500:W = 34: GOSUB 500
700 GOTO 610
800 W = 29: GOSUB 500
805 PRINT "THAT ";
810 W = 30: GOSUB 500
815 PRINT "IS ";
820 W = 31: GOSUB 500
825 PRINT "CORRECT"
830 FOR R = 1 TO 200: NEXT
850 N = A: GOSUB 100
855 PRINT : PRINT A;
860 W = 36: GOSUB 500
865 PRINT " + ";
870 N = B: GOSUB 100
875 PRINT B;
880 W = 30: GOSUB 500
885 PRINT " IS ";
890 N = A + B: GOSUB 100
895 PRINT A + B
897 FOR R = 1 TO 150: NEXT R
900 NEXT C
910 FOR D = 1 TO 300: NEXT
915 PRINT "YOU ";:W = 44: GOSUB 500
917 PRINT "GOT ";:W = 45: GOSUB 500
919 PRINT NR;" ";:N = NR: GOSUB 100
921 PRINT "OUT ";:W = 46: GOSUB 500
923 PRINT "OF ";:W = 47: GOSUB 500
925 PRINT P;" ";:N = P: GOSUB 100
927 PRINT "CORRECT ";:W = 31: GOSUB 500
929 PRINT : PRINT "OR ";:W = 49: GOSUB 500
931 PRINT INT ((NR / P) * 100);" ";:N = INT ((
    NR / P) * 100): GOSUB 100
935 PRINT "PERCENT":W = 50: GOSUB 500
940 END
1000 DATA 39,WELCOME,2,TO,40,MATH,41,QUIZ,1,ONE

1010 DATA 51,HOW,52,MANY,48,PROBLEMS
1020 FOR R = 1 TO 5: READ W,A$
1030 GOSUB 500
1040 PRINT A$;" ";
1045 FOR D = 1 TO 130: NEXT D
1050 NEXT
1055 PRINT : PRINT : FOR D = 1 TO 300: NEXT D
1060 FOR R = 1 TO 3: READ W,A$: GOSUB 500
1065 FOR D = 1 TO 130: NEXT D
1070 PRINT A$;" ";: NEXT
1080 INPUT P
1090 N = P: GOSUB 100
1100 RETURN
```

Program 4.

```
5 SP = 10
7 DIM W$(65)
100 TEXT : HOME
110 HIMEM: (5 * 4096) - 1: REM $4FFF
112 REC = 9 * 4096: REM   $9000
113 PLAY = 9 * 4096 + 4 * 16 + 9: REM  $9049
115 IF  PEEK (REC) = 32 THEN 140
120 PRINT CHR$ (4);"BLOAD TABLE,A$310"
130 PRINT CHR$ (4);"BLOAD VOC 1.1.OBJ0,A$9000"

140 HTAB 10
150 PRINT "VOCABULARY BUILDER"
160 POKE 34,1
170 VTAB 5
180 PRINT "HAVE YOU ALREADY MADE A VOCABULARY 
    ?";
185 GOSUB 5000
190 IF F = 0 THEN 260
200 PRINT : PRINT : PRINT "HIT A KEY";
210 GET T$
215 PRINT
220 PRINT CHR$ (4);"CATALOG"
230 INPUT "TYPE YOUR FILENAME AND HIT RETURN 
          (RET FOR NONE)===>";N$
240 IF N$ = "" THEN 260
250 PRINT CHR$ (4);"BLOAD ";N$;",A$5000"
252 PRINT CHR$ (4);"OPEN";N$;".VOC"
253 PRINT CHR$ (4);"READ";N$;".VOC"
254 FOR R = 0 TO 64
255 INPUT W$(R)
256 NEXT R
257 PRINT CHR$ (4);"CLOSE"
260 REM  MAIN MENU
270 HOME
280 HTAB 15: PRINT "MAIN MENU"
290 VTAB 7
300 PRINT "1-ENTER A WORD
310 PRINT
320 PRINT "2-PLAY A WORD
330 PRINT
340 PRINT "3-PRINT A VOCABULARY SHEET"
350 PRINT
360 PRINT "4-QUIT"
370 PRINT : PRINT
380 PRINT "ENTER YOUR SELECTION==>";
390 GET C$
400 IF C$ < "1" OR C$ > "4" THEN 390
410 PRINT C$
420 ON VAL (C$) GOTO 1000,2000,3000,430
430 PRINT : PRINT "DO YOU REALLY WANT TO QUIT 
    ";
440 GOSUB 5000
450 IF F = 0 THEN 260
460 FOR R = 1 TO 20: PRINT : NEXT
470 INPUT "ENTER FILENAME TO SAVE AND HIT RETU
    RN  (RET FOR NONE)";N$
480 IF N$ = "" THEN 30000
490 PRINT CHR$ (4);"BSAVE";N$;",A$5000,L$3FFF"


500 PRINT CHR$ (4);"OPEN";N$;".VOC"
510 PRINT CHR$ (4);"WRITE";N$;".VOC"
520 FOR WO = 0 TO 64
530 PRINT W$(WO)
540 NEXT WO
550 GOTO 30000
1000 HOME
1010 PRINT "SINGLE WORD OR SERIES (S OR E)?";
1020 GET T$
1025 PRINT
1030 IF T$ = "S" THEN 1090
1050 PRINT : PRINT "ENTER STARTING WORD NUMBER 
     ";
1060 INPUT ST
1070 INPUT "ENDING WORD NUMBER ";EN
1080 GOTO 1100
1090 INPUT "ENTER WORD NUMBER ";ST:EN = ST
1100 FOR WO = ST TO EN
1110 HOME
1120 PRINT "WORD NUMBER :";WO
1130 VTAB 5
1140 PRINT "ENTER WORD NAME - DEFAULT=";W$(WO)
1150 INPUT N$
1160 IF N$ = "" THEN N$ = W$(WO)
1170 W$(WO) = N$
1180 VTAB 10
1190 PRINT "ENTER SPEED - DEFAULT=";SP
1200 INPUT N$
1210 IF N$ = "" THEN 1230
1220 SP = VAL (N$)
1230 POKE 772,SP
1240 POKE 0,WO
1250 PRINT : PRINT "HIT ANY KEY TO RECORD"
1260 GET T$
1270 CALL REC
```

```
1280 PRINT : PRINT "HIT ANY KEY FOR PLAYBACK"
1290 GET T$
1300 CALL PLAY
1310 PRINT "WAS THAT OK ?";: GOSUB 5000
1320 IF F = 0 THEN 1110
1330 NEXT WO: GOTO 260
2000 HOME
2010 PRINT "SINGLE WORD OR SERIES (S OR E)?";
2020 GET T$
2030 PRINT
2040 IF T$ = "S" THEN 2090
2050 PRINT : PRINT "ENTER STARTING WORD NUMBER -
   ";
2060 INPUT ST
2070 INPUT "ENDING WORD NUMBER ";EN
2080 GOTO 2100
2090 INPUT "ENTER WORD NUMBER ";ST:EN = ST
2100 FOR WO = ST TO EN
2110 HOME
2120 PRINT "WORD NUMBER :";WO
2130 VTAB 5
2140 PRINT "ENTER WORD NAME - DEFAULT=";W$(WO)
2150 INPUT N$
2160 IF N$ = "" THEN N$ = W$(WO)
2170 W$(WO) = N$
2180 VTAB 10
2190 PRINT "ENTER SPEED - DEFAULT=";SP
2200 INPUT N$
2210 IF N$ = "" THEN 2230
2220 SP = VAL (N$)
2230 POKE 772,SP
2240 POKE 0,WO
2280 PRINT : PRINT "HIT ANY KEY FOR PLAYBACK"
2290 GET T$
2300 CALL PLAY
2330 NEXT WO: GOTO 260
2670 CALL REC
3000 HOME
3005 HTAB 5
3010 PRINT "HIT ANY KEY TO START PRINTOUT"
3020 PRINT CHR$ (4);"PR#1"
3030 PRINT "WORD NUMBER"; TAB( 20);"WORD"
3040 FOR X = 1 TO 40: PRINT "-";: NEXT X
3045 PRINT
3050 FOR WO = 0 TO 63
3060 PRINT WO; TAB( 20);W$(WO)
3070 NEXT WO
3075 PRINT CHR$ (4);"PR#0"
3080 GOTO 260
4999 END
5000 GET T$
5010 IF T$ < > "Y" AND T$ < > "N" THEN 5000
5020 F = 0
5030 IF T$ = "N" THEN PRINT "NO"
5040 IF T$ = "Y" THEN F = 1: PRINT "YES"
5050 RETURN
30000 END
```

©

Function VAL (X) In UCSD PASCAL For Apple II

Michael Erperstorfer
Vienna, Austria

Function VAL (X) is similar to BASIC's VAL-function:
X must be a string of an integer number;
VAL returns a true integer number;
If X is no integer number VAL returns 0;
String X may have leading or trailing spaces.

```
PROGRAM VALTEST;
VAR INPUT : STRING;
FUNCTION VAL (S : STRING) : INTEGER;
 VAR START,I,LEN,O,V : INTEGER;
     NEG : BOOLEAN;
 BEGIN
  V:=0;
  NEG:=FALSE;
  WHILE COPY (S,1,1) = ' ' DO S:=COPY (S,2,
    LENGTH (S)-1);
  (* remove blanks from left *)
  WHILE COPY (S,LENGTH (S),1)=' ' DO S:=COPY
    (S,1,LENGTH (S)-1);
  (* remove blanks from right *)
  START:=1;
  IF COPY (S,1,1) = '-' THEN
   BEGIN
    START:=2;
    NEG:=TRUE
   END;
   (* if first char = '-' *)
   (* number is negative *)
   (* increment start value *)
   (* to skip '-' sign *)
   (* set neg-flag *)
  LEN:=LENGTH (S);
  FOR I:=START TO LEN DO
   BEGIN
    O:=ORD (S[I]);
    IF (O>47) AND (O<58) THEN
    (*check if char is number *)
     V:=V + TRUNC (PWROFTEN (LEN-I)) * (O-48)
     (* calculate value *)
    ELSE
     BEGIN
     (* if char is not number *)
      VAL:=0;
      (* set value to 0 *)
      EXIT (VAL)
      (* and exit function *)
     END
   END;
  IF NEG THEN VAL:=-V ELSE VAL:=V
 END;
BEGIN
 REPEAT
  WRITE ('STRING: ');
  READLN (INPUT);
  WRITELN ('=',VAL (INPUT))
 UNTIL INPUT=''
END.
```

©

To check a tape using this program, rewind the tape after a SAVE (while the program is still also in the computer's memory), type CALL 768, and do not hit return until after *you have started your tape.*

Verify Your Applesoft Tapes

Keith Falkner
Venice, FL

Imagine this – you've written a dandy program in Applesoft, tested it, debugged it, perfected it, and of course SAVEd it.

But is the program *really* saved? Can you load the tape? If the tape recorder has developed a problem, you may lose this program forever as soon as you type NEW or turn off your Apple.

Here is how to know for sure. Below is a machine language program which verifies the accuracy of a SAVEd Applesoft program on tape. To make use of this program:

1. Type in Program 1 and RUN it.

2. From the machine language monitor, SAVE it to tape via 300.393W or to disk via BSAVE VERIFY,A$300,L$94.

3. When you need it, BLOAD VERIFY from disk or enter the monitor with CALL -151 and reload it from tape via 300.393R (this does *not* affect an Applesoft program in memory).

4. SAVE the Applesoft program as normal.

5. Operate the tape recorder just as you would to LOAD an Applesoft program, but type CALL 768 instead of LOAD. The tape will be read and compared to the Applesoft program.

6. If the comparison is successful, there will be no error message, just the two BEEPs which accommmpany LOAD.

7. If, alas, the tape is not a readable copy of the program, the message ERR will appear, with the address of the error and the values of the byte on tape and the byte in memory.

An error message is never good news, but it is far better to know of a problem before the program is lost than to rely on a tape which later proves unreadable.

An Applesoft program on tape is really two data records: the first record is four bytes long and indicates the size of the Applesoft program. If this header is read accurately, the computer beeps, but prints nothing. The second data record on tape is as long as the header indicates, and contains an image of the program. When this is successfully read, whether by LOAD or by the verify program below, the computer beeps again.

Load naturally shoves the incoming data into memory, but Program 1 harmlessly compares what is read with what is in memory. If those bytes differ, an error message appears: ERR 08EB-88 (8C) for example, which means that at location $8EB, the byte in memory is $88 (the token for GR), but the tape contains $8C (the token for CALL). As soon as it reports such an error, the VERIFY routine quits. At this point, nothing in memory has been altered, so the SAVE can be retried, perhaps with a different tape or a different volume level.

Take the time to type this routine into your Apple and save it. Sooner or later you will want assurance that a saved Applesoft tape is the accurate program you hope it is.

Type in the Applesoft program and it will build this machine language verify routine starting at address 768 when you type RUN.

```
100 FOR I=768 TO 915:READ X:POKE I,
      X:NEXT
768 DATA 162, 0, 32, 117, 253, 160, 2,
      138, 145, 105
778 DATA 200, 169, 0, 145, 105, 200, 169,
      2, 145, 105
788 DATA 189, 9, 2, 41, 127, 157, 0,
      2, 202, 224
798 DATA 255, 208, 243, 96, 32, 61, 3,
      165, 103, 133
808 DATA 60, 165, 104, 133, 61, 165, 175,
      133, 62, 165
818 DATA 176, 133, 63, 32, 61, 3, 169,
      141, 76, 237
828 DATA 253, 32, 250, 252, 169, 22, 32,
      201, 252, 133
838 DATA 46, 32, 250, 252, 160, 36, 32,
      253, 252, 176
848 DATA 249, 32, 253, 252, 160, 59, 32,
      236, 252, 240
858 DATA 14, 69, 46, 133, 46, 32, 186,
      252, 160, 52
868 DATA 144, 240, 76, 38, 255, 234, 234,
      234, 193, 60
878 DATA 240, 235, 72, 32, 45, 255, 32,
      146, 253, 177
888 DATA 60, 32, 218, 253, 169, 160, 32,
      237, 253, 169
898 DATA 168, 32, 237, 253, 104, 32, 218
      , 253, 169, 169
908 DATA 32, 237, 253, 169, 141, 76, 237
      , 253
```

©

DISK DRIVE FOR APPLE II*

100% Apple compatible; runs DOS 3.3, PASCAL, CP/M etc. 15% more storage by using enhancer diskette. 300% faster track-to-track speed	$259.00

Drive can be used with other computers by simply disconnecting interface board.

APPLE II + Complete System: APPLE II + Computer, disk drive and controller 12" monitor green screen	**$1599.00**

APPLE II + compatible Winchester drives - 5 MB . . . $1795   10 MB . . . $1995
Includes controller, cables, software for CP/M, DOS or PASCAL.

8" drive, controller, power supply, cables, cabinet and software	**$1595.00**

OTHER PRODUCTS FOR APPLE II

Description	Sale Price
VERSAcard-multifunction board	$169.00
BSR Transducer	$19.00
Applications/Demo Disk	$25.00
Parallel Printer Card	$69.00
PRT-1 with cable	$75.00
PRT-1 with EPSON 80/100 screen graphics dump with cable -Graffitti card-	$99.00
PRT-1 with NEC8023 or C-10th PROWRITER screen Dump Graphics	$99.00
16K ram card	$89.00
32K ram card	$175.00
64K ram card including DOS 3.3 disk emulator	$279.00
128K ram card including DOS 3.3 disk emulator	$369.00
PASCAL disk emulator	$39.00
DOS 3.3 disk emulator	$39.00
Visicalc Expand Program	$59.00
VERSAbox Spooler/buffer 16K Centronics Input/Output	$199.00
VERSAbox Spooler/buffer 16K Centronics and RS232C	$239.00
VERSAbox Real Time Clock/ Display option	$129.00
16K Memory Modules for VERSAbox	$39.00
Standard 6' Centronics Parallel Cable for EPSON, C-ITOH, NEC, ANADEX, and others	$22.00
Standard 6' Paper Tiger or Prism Printer cable. Parallel Centronics Male DB25	$26.00
Miscellaneous cables for various printers. Please specify for Qume, Diable, Votrax NEC Spinwriter or other	$26.00
80 Column card	$169.00
Z80 card (no CP/M software included)	$149.00
Controller for Apple II including hardware diagnostics	$95.00

PRINTERS AND MONITORS

BMC 12" Monitor (green screen)	$95.00
BM12EN 12" (green screen) high res	$149.00
BMC1400 12" CLU COLOR Monitor	$289.00
8510AP Pro-Writer (Parallel, dot matrix)	$499.00
8510ACD Pro-Writer (Serial, dot matrix)	$649.00
F10 Daisey Wheel Printer (Letter quality)	$1495.00
1550AP 15" Pro-Writer II (Parallel, dot matrix)	$699.00
1550CD 15" Pro-Writer II (Serial, dot matrix)	$849.00

Soon opening retail stores throughout USA
• Franchise inquiries welcome •
PRICES SUBJECT TO CHANGE WITHOUT NOTICE

BASIS & FRANKLIN Computers available

COMPUSHACK
Computers - Video Games
2630-H Walnut Avenue
Tustin, CA 92680

• 24 HOUR ORDERING SERVICE •
TOLL-FREE TELEPHONE NUMBERS:
Outside California: (800) 854-8426
Inside California: (714) 730-7207 or (408) 973-1444
Telex: 18-3511 Answer Back CSMA

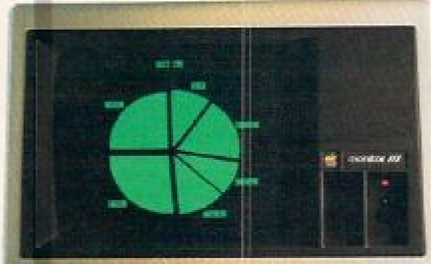
*Apple II is a trademark of Apple Inc.

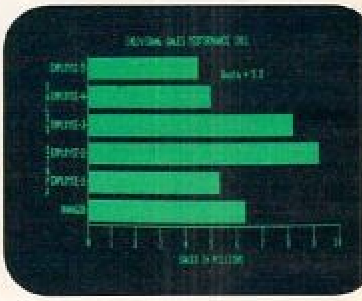
Not all business
And we've got the

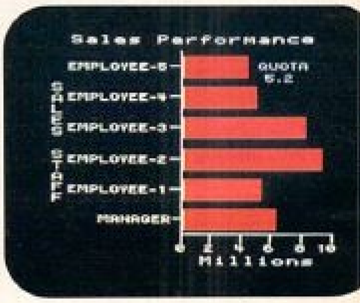
As you know, one picture is worth a few thousand numbers.

As you may not know, Apple® Business Graphics software can generate more types of pictures, in more colors, using more data than any other graphics package.

So you not only get the usual bar graphs and pie charts. You also get unusual bar graphs and pie charts. Sophisticated line and area graphs. Even scattergrams. All teamed with extremely useful and powerful features—exploded views, unlimited overlays, floating titles and more.

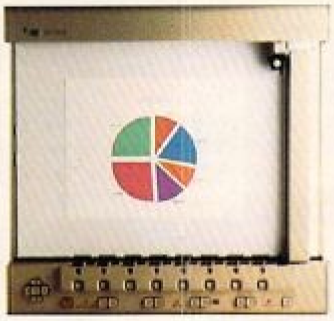


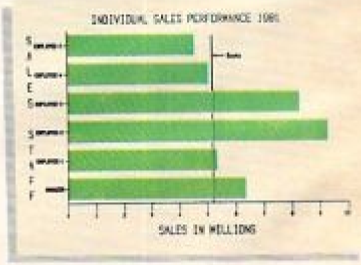




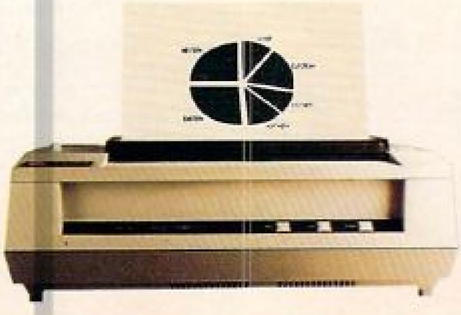
	Apple	VisiTrend/ VisiPlot	pfsGraph
Graph Types			
Line	Yes	Yes	Yes
Vertical Bar	Yes	Yes	Yes
Horizontal Bar	Yes	No	No
Side-by-side Bar	Up to 4	2	4
Pie	Yes	Yes	Yes
Partial Pie	Yes	No	No
Scattergram	Yes	Yes	No
Curve Fitting	5 Kinds	1	None
Data Points (Max.)	3500+	645	36
Plotter Compatible	Virtually Any	None	H-P7470A Only
Compatible File Types	Pascal BASIC VisiCalc	BASIC VisiCalc	pfs VisiCalc
Math Functions	Yes	Yes	No
Available Colors	6	4	4

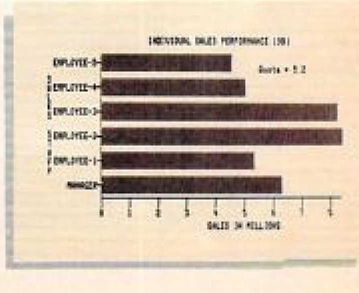
Apple Business Graphics is available for both the Apple II and Apple III.



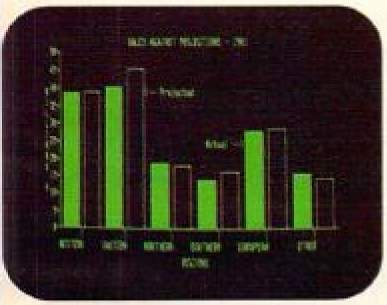


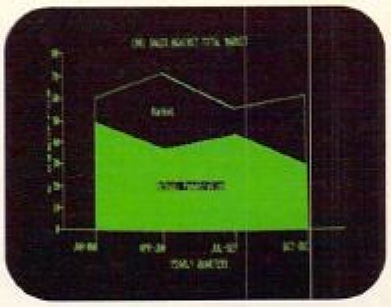
Equally important, with our graphics package you'll find more ways to see what you're doing. On the monitor of your choice. And on virtually any printer or plotter on the market.

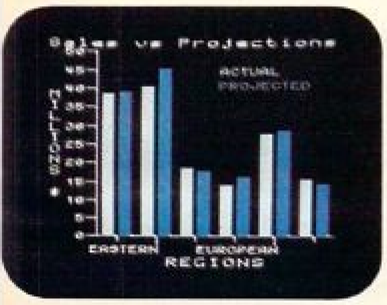


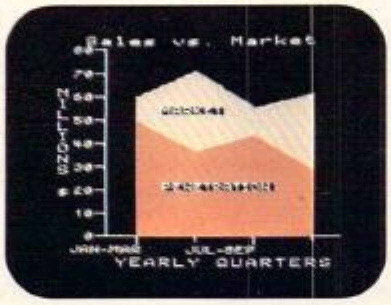


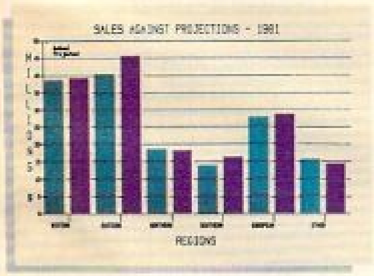
graphics are alike.
pictures to prove it.

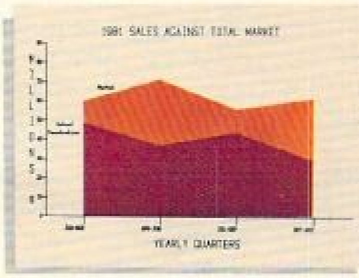


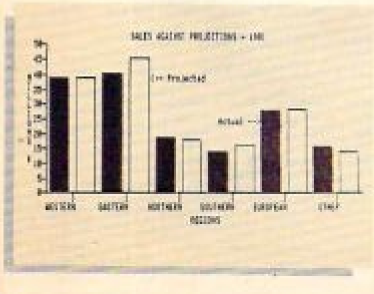


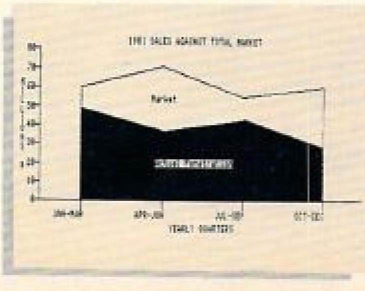












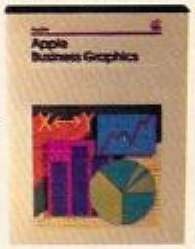
Even on transparencies and slides (by combining Apple Business Graphics with packages like Screen Director™ and Target Image Maker™).

All of which makes for more presentable presentations. And more revealing market analyses, forecasts, budgets, stock trends, business plans or customer demographics.

Or the information of your choice from the files of your choice. Be it VisiCalc®, Pascal, DIF or BASIC.

We could easily tell you more.

But we'd rather show you more. In person. At any of our over 1300 full-support dealers (they also offer a vast library of other quality software distributed by Apple for Apples).



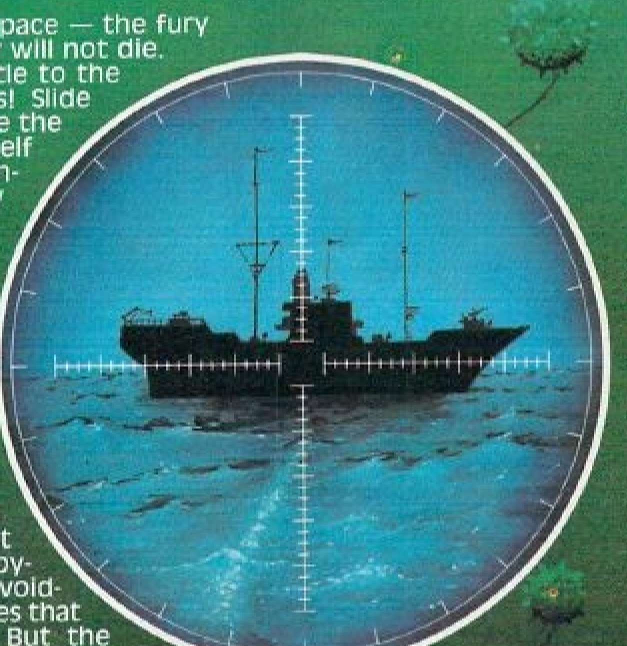
So pay one a visit. And find out how easy it is to turn a sea of data into data you can see.

apple®

The most personal software.

Call (800) 538-9696 for the location of the authorized Apple dealer nearest you, or for information regarding corporate purchases through our National Account Program. In California (800) 662-9238. Or write Apple Computer Inc., Advertising and Promotion Dept., 20525 Mariani Ave., Cupertino, CA 95014.

ABSOLUTELY RELENTLESS



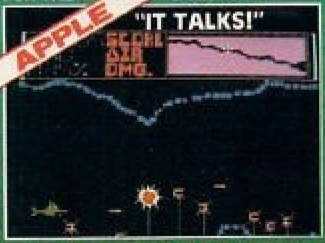
The challenge of inner space — the fury of an enemy that seemingly will not die. This is SEA DRAGON — a battle to the death **under** the high seas! Slide into the Captain's chair, take the controls and prepare yourself for the most incredible non-stop action this side of Davy Jones' locker. SEA DRAGON puts you in control of a nuclear sub that's armed from stem to stern with enough firepower to take on King Neptune himself — and you'll need every missile, every torpedo, and every scrap of skill you can muster to survive.

The object of SEA DRAGON is to successfully navigate your sub through an underwater course past mountains and through labyrinthine passageways while avoiding clusters of explosive mines that rise from the seabottom. But the danger doesn't stop there — overhead, surface destroyers lace the water with depth charges; below, deadly attack bases and arcing lasers cut a killing swath that could reduce your sub to bubbling slag. But even these potentially lethal perils are dwarfed by the awesome menace that awaits you at the course's end.

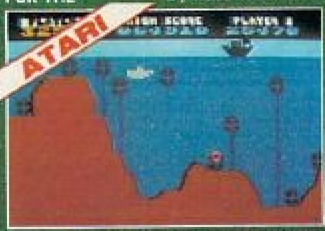
SEA DRAGON — every possible "extra" is here to ensure your playing pleasure: exciting sounds, high score save, machine language graphics and an eye-popping scrolling seascape that extends the equivalent of over two dozen screens placed end-to-end, providing a diverse and unique challenge that will not diminish after repeated playings.

Nothing you've ever seen on your micro could possibly prepare you for this! You are ready now, ready for the ultimate in undersea action with a pace that is absoutely unyielding. SEA DRAGON — the arcade has finally come home.

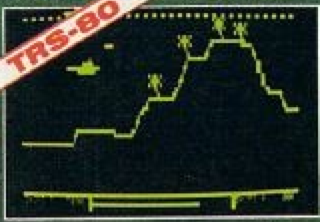
FOR THE APPLE by John Anderson



FOR THE ATARI by Russ Wetmore



FOR THE TRS-80 by Wayne Westmoreland & Terry Gilman



ALL VERSIONS ARE JOYSTICK COMPATIBLE!

SEA DRAGON FEATURES

- Fantastic Scrolling Seascape
- Nearly Limitless Game Challenge
- High Score Save (disk version)
- Terrific Sound Effects
- Arcade Action Graphics™
- Apple version "talks" without special hardware!

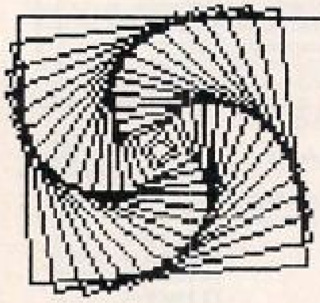
ORDERING INFORMATION

APPLE 2 or APPLE 2 PLUS 48K Disk (DOS 3.3 required)	
042-0146	$34.95
ATARI 32K Disk	
052-0146	$34.95
ATARI 16K Tape	
051-0146	$34.95
TRS-80 32K Disk	
012-0146	$24.95
TRS-80 16K Tape	
010-0146	$24.95

Adventure International
A DIVISION OF SCOTT ADAMS, INC.
PRICES SUBJECT TO CHANGE

To order, see your local dealer. If he does not have the program, then call 1-800-327-7172 (orders only please) or write for our free catalog.

Published by ADVENTURE INTERNATIONAL
a subsidiary of Scott Adams, Inc.
BOX 3435 • LONGWOOD, FL 32750 • (305) 830-8194



A Monthly Column

Friends Of The Turtle

David D. Thornburg
Associate Editor

Recursion – Part I

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

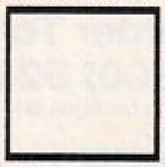
In computer languages, recursion can take several forms. Recursion is probably the most powerful and least understood programming tool in existence. Because LOGO is a marvelous language for exploring this topic, *and* because recursion can let you generate some beautiful pictures with programs only a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

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

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

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

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



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

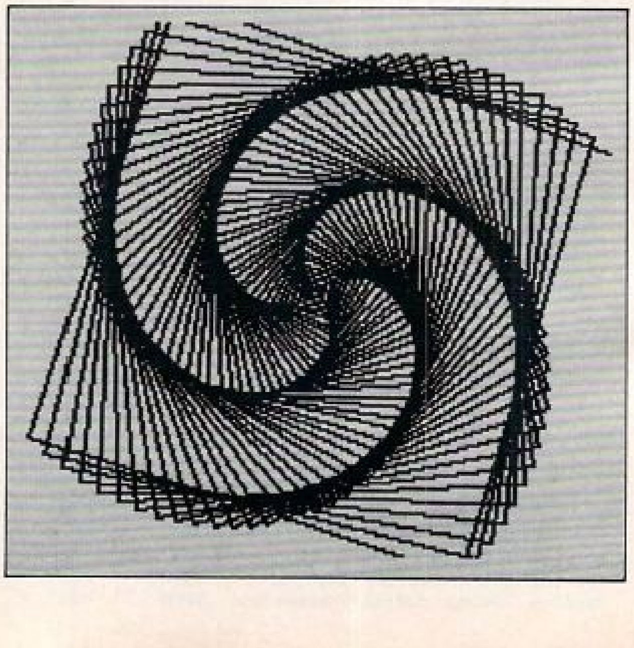
This type of tail-end recursion is available in languages like PILOT through the use of the jump (J:) command, or in BASIC through the GOTO command. The equivalent SQUARE procedure in Atari PILOT looks like this:

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

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

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

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



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

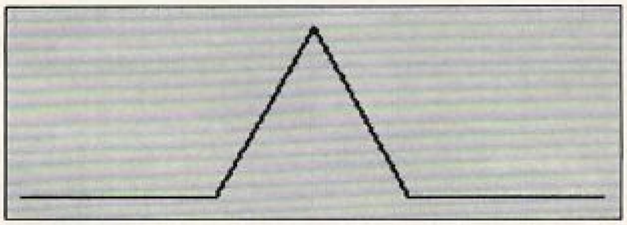
The SQUIRAL procedure also repeats forever, but it does not retrace its own path. This type of tail-end recursion is also possible in languages that have only global (rather than local) variables. In Atari PILOT, for example, this procedure would look like this:

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

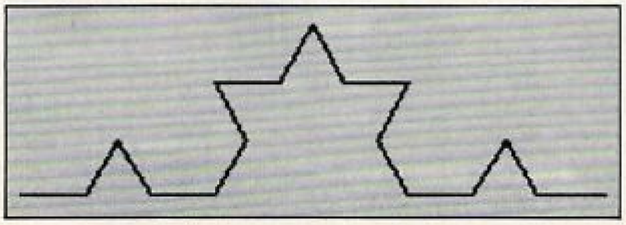
The use of the compute (C:) command allows us to change the value of the numeric variable #S.

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

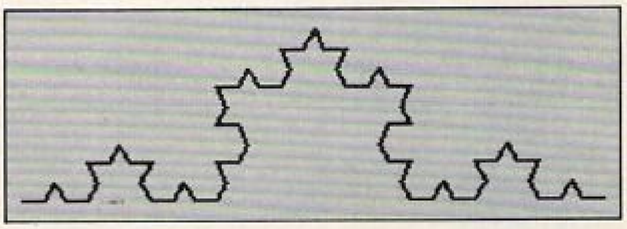
In order to show how general recursion works, we will explore some curves that we described a few columns back – the fractals. Fractal curves are generated by the continued repetition of a simple motif within each portion of an overall curve. For example, suppose we start with the same curve we used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



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



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



Explicit Procedures For Drawing Fractals

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

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

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

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

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

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

Now enter:

```
CLEARSCREEN
SETUP
K1 243
```

You should see the first level curve on the screen.

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

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

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

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

```
CLEARSCREEN
SETUP
K2 243
```

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

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

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

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

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

Recursive Procedures For Drawing Fractals

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

Table of Command Sequences for K2

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

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

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

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

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

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

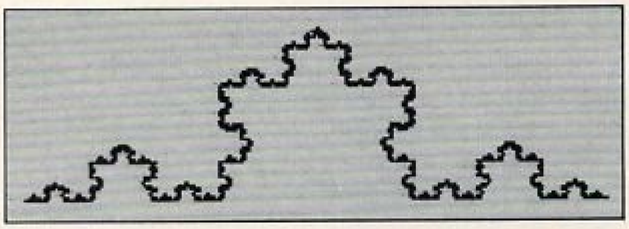
To see how this procedure operates, let's examine it line by line. Suppose you were to give the command TRIAD 243 100, for example. First, the size (243) would be tested to see if it was less than the limit (100). Because it is not, TRIAD would be used again with a size of 243/3, or 81. Since in this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

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

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

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

```
CLEARSCREEN
SETUP
TRIAD 243 3
```



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

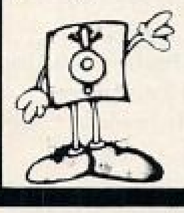
Friends of the Turtle
P.O. Box 1317
Los Altos, CA 94022

©

Scotch Diskettes

Rely on Scotch® diskettes to keep your valuable data safe. Dependable Scotch diskettes are tested and guaranteed error-free. The low abrasivity saves your read/write heads. They're compatible with most diskette drives.

(800)235-4137



VISA

Dealer Inquiries Invited

PACIFIC EXCHANGES
100 Foothill Blvd.
San Luis Obispo, CA
93401. In Cal. call
(800) 592-5935 or
(805) 543-1037

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

Apple Menu

Robert J. Beck
Minneapolis, MN

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

An Alphabet Menu

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

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

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

An Arrow Menu

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

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

Let's take it one line at a time. Line 10 initializes some important variables: HT (horizontal tab) is the number of spaces that the option list is tabbed over, VT (vertical tab) is the vertical line number at which the list begins, N is the number of options, and T is used to keep track of which choice the arrow points at. Line 20 prints the title, line 30 tabs to the preset vertical line, and lines 40-80 print the menu. The first POKE in line 90 sets line width to three spaces; the second POKE sets the left margin five spaces to the left of the menu. A VTAB to the top of the menu list completes the preparations for printing the arrow and GETting a keypress at line 110.

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

Written for the Apple II and Apple II Plus, this article explains the sounds of the Apple, with special attention to its Music routine.

Apple Sounds

Michael P. Antonovich
Wyomissing, PA

When you are writing software for the Apple II, are you afraid to add sound to your program because you feel that it is too hard or that you have a tin ear? I have tested many Apple programs, and those which incorporate sound, quality graphics, and user-friendly input are often the ones which stand out in my mind. If you haven't been using sound, read on and see how easy Apple sounds really are.

When you bought your Apple II, you received a chip called the Programmer's Aid already plugged into your motherboard. This chip (a ROM – Read Only Memory) contains a group of utilities for the Integer BASIC program. Some of you probably have the Apple II Plus rather than an Apple II. However, don't stop reading if you bought the Language Card.

Maybe you have never looked at what the Programmer's Aid does. If you have, you noticed a utility called the *Music* routine. The Music routine lets you create music with the options of changing the note's pitch, duration, and timbre.

To play a note, you have to POKE three items into memory before you can call the Music routine. The first item is the timbre. Timbre will make the same note sound slightly different, but you have only five possible timbre selections. They are: 2, 8, 16, 32, and 64. The timbre you want must be POKEd into decimal memory location 765 as follows:

POKE 765,32

Timbre 32 will give you the cleanest notes over the Apple's range.

The second item you need to store is the note's duration. A decimal value from 1 to 255 can be POKEd into decimal address 766 to produce short to long notes, respectively. A value of 170 will result in a note of approximately one-second duration. The POKE command is:

POKE 766,170

Third, you must select the note which you want. The note decimal values can range from 1 to 50, with 1 being the low end of the scale. The numbers are based on a chromatic scale. Values of the notes above 50 will result in a random note selection. Middle C corresponds roughly to the POKE:

POKE 767,32

Finally, you are ready to play the note. To do this, you must call the machine language routine located at decimal address -10473 or 55063, using a CALL statement as in:

CALL -10473

There! How did that sound? Just one note, you say? Well, the following program lists about a dozen sounds made using the above techniques. They are by no means the limits of the Apple's ability. Rather, they are presented to whet your appetite so that you will try to create some sounds on your own.

```
9 REM  FALLING SOUND
10 POKE 766,2: POKE 765,32: FOR I=50 TO 1 STE
    P -1: POKE 767,I: CALL-10473:NEXTI
11 END
19 REM RISING SOUND
20 POKE 766,2: POKE 765,32: FOR I=1 TO 50: PO
    KE 767,I: CALL -10473: NEXT I
21 END
29 REM VARIOUS WHIRLING SOUNDS
30 POKE 766,2: POKE 765,32: FOR I=20 TO 30: P
    OKE 767,I: CALL -10473: NEXT I
31 FOR I=50 TO 10 STEP -1: POKE 767,I: CALL -
    10473: NEXT I: GOTO 30
40 POKE 766,2: POKE 765,32: FOR I=20 TO 30: P
    OKE 767,I: CALL -10473: NEXT I
41 FOR I=30 TO 20 STEP -1: POKE 767,I: CALL -
    10473: NEXT I: GOTO 40
50 POKE 766,2: POKE 765,32: FOR I=10 TO 40: P
    OKE 767,I: CALL -10473: NEXT I
51 FOR I=30 TO 20 STEP -1: POKE 767,I: CALL -
    10473: NEXT I: GOTO 50
59 REM WARNING SIREN
60 POKE 766,2: POKE 765,32: FOR I=1 TO 100: P
    OKE 767,40: CALL -10473: NEXT I
61 POKE 766,4: FOR I=30 TO 20 STEP -1: POKE 7
    67,I: CALL -10473: NEXT I
62 FOR I=10 TO 40: POKE 767,I: CALL -10473 NE
    XT I: GOTO 60
64 REM LIGHT SABER
65 POKE 766,2: POKE 765,32: FOR I=1 TO 100: P
    OKE 767,1: CALL -10473: NEXT I
66 FOR I=10 TO 40: POKE 767,I: CALL -10473: N
    EXT I
67 FOR I=30 TO 20 STEP -1: POKE 767,I: CALL -
    10473:  NEXT I: GOTO 65
69 REM   PADDLE 0 CONTROLLED MOTOR
70 POKE 766,2:POKE 765,32:POKE 767,140:CALL-1
    0473:FOR I=1 TO PDL(0):NEXTI:GOTO70
79 REM  PADDLE CONTROLLED SOUNDS
80 POKE 766, PDL (1): POKE 765,32: POKE 767,
    PDL (0): CALL -10473: GOTO 80
89 REM  RANDOM NOISE
90 D=64
91 POKE 766,2:POKE 765,D:POKE 767,100:CALL -1
    0473:D=D/2:IF D<1 THEN D=64:GOTO91
99 REM  ALIEN SPACESHIP SOUNDS
100 D=64
101 POKE 766,2:POKE 765,D:POKE 767,150:CALL-10
    473:D=D/2:IF D<1THEN D=64:GOTO101
110 D=2
111 POKE 766,2:POKE 765,D:POKE 767,150:CALL-10
    473:D=D*2:IF D>64 THEND=2:GOTO111
199 REM  MORE SOUNDS
210 POKE 766,2:POKE 765,32:FORI=50TO1STEP-1:PO
    KE767,I:CALL-10473:NEXT I:GOTO210
220 POKE 766,2:POKE 765,32:FOR I=1 TO 50:POKE
    767,I:CALL -10473:NEXT I:GOTO 220
999 END
```

©

Just when the business world is up to its white collars in visiclones, calcalikes and other spreadsheet packages, Apple's® come out with something entirely different.

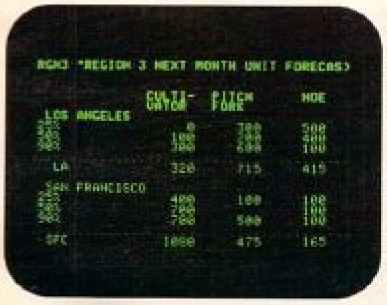
Introducing Senior Analyst.

Like other financial modeling packages, it allows managers and professionals to ask all those proverbial "What If?" questions.

Unlike the others, this powerful financial planning tool was designed to be used in a corporate environment, by lots of people. So you get lots of advantages.

For example, you can transfer data (across diskettes) from one financial model to another. Or consolidate many models into one.

So sales, manufacturing, administration and any number of other departments (even in other cities) can easily share information. Giving each the power to create comprehensive and



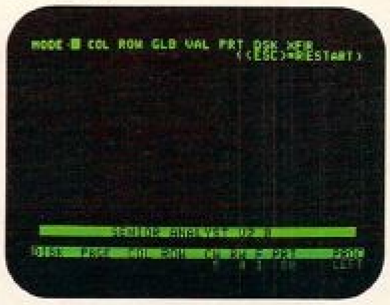
```
RGN3 "REGION 3 NEXT MONTH UNIT FORECAS>

                 CULTI-  PITCH      HOE
                 VATOR   FORK
 LOS ANGELES
 [illegible]         0     300      500
                   100     200      400
                   300     600      100
   LA              320     715      415
 SAN FRANCISCO
 [illegible]       400     100      100
                   700              100
                   700     500      100
  SFC             1000     475      165
```

With Senior Analyst, you can now cultivate forecasts by merging reports from distant divisions.

flexible financial projections, budgets, cash flow statements and the like.

Want to combine selected data (such as important subtotals) from six different divisions? With Senior Analyst, you can do it. And even print out a formatted report that includes only the information you need.

A report that anyone can understand. Because the headings are in English, not in code.



```
MODE   COL ROW GLB VAL PRT DSK XFR
                          (<ESC>=RESTART)



          SENIOR ANALYST V2.0
DISK PAGE COL ROW CW RW F PRT        PROC
                                     LEFT
```

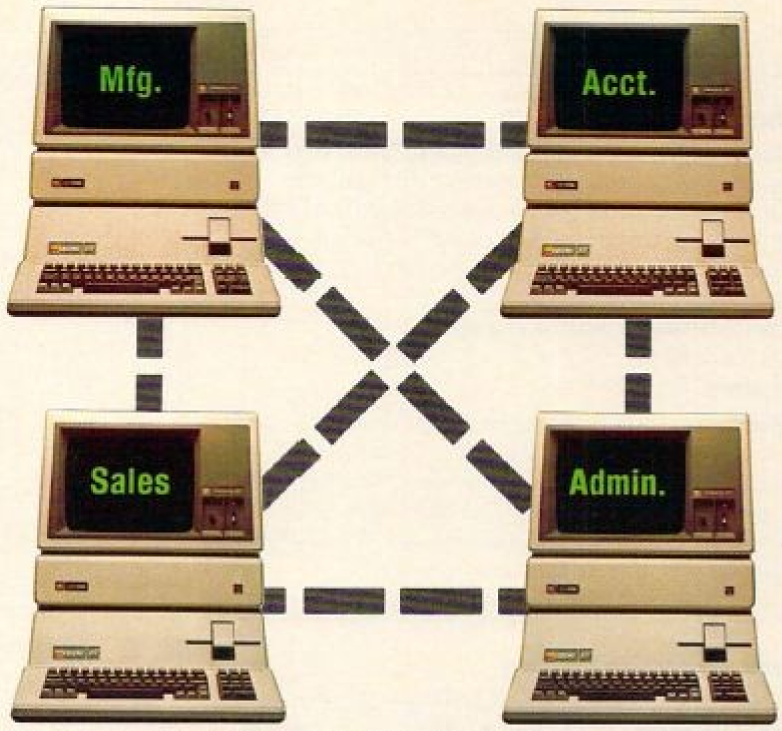
Easy to follow commands allow employees to create models without learning a second language.

You can also document and print out all those assumptions used to create your model, to give others a concrete understanding of how you reached your conclusion. (The program even allows you to continue working while a model is being printed.)

To complement all these accommodating features, you'll also find built-in functions for depreciation, linear regression forecasting, and other powerful virtues not found in most financial software packages.

All of which we'd like you to experience in person, at any of our 1300 authorized full-support dealers (they also offer a vast library of other quality software distributed by Apple for Apples).

And don't ask for just any spreadsheet package. Tell them you need to see an analyst.



Senior Analyst lets different departments share and consolidate data. That way the company doesn't make more pitchforks than it can sell.

apple®

The most personal software.

Call (800) 538-9696 for the location of the authorized Apple dealer nearest you, or for information regarding corporate purchases through our National Account Program. In California (800) 662-9238. Or write Apple Computer Inc., Advertising and Promotion Dept., 20525 Mariani Ave., Cupertino, CA 95014. © 1982 Apple Computer Inc.

BOOST YOUR APPLE® POWER

With Wiley's Friendly Paperback Guides

BASIC FOR THE APPLE II®

Jerald R. Brown, LeRoy Finkel, & Bob Albrecht

A complete, friendly, and virtually guaranteed introduction to BASIC programming on the Apple II®—an Applesoft® version of the authors' *BASIC For Home Computers*, which *Microcomputing* magazine says "takes you from knowing almost nothing to knowing almost everything."

(1-86596-6) November 1982
$12.95

6502 ASSEMBLY LANGUAGE PROGRAMMING

Judi N. Fernandez, Donna N. Tabler, & Ruth Ashley

Now even computer novices can take advantage of the speed and efficiency of assembly language programming—with the help of this easy-to-use, self-paced guide.

(1-86120-0) November 1982
$12.95

APPLE II® PROGRAMMING EXERCISES

Leo J. Scanlon

This exciting workbook gives you the hands-on exercises and problems you need to perfect your assembly language skills on the Apple II®...and become adept at writing programs for processing lists, input/output, musical compositions, and more.

(1-86598-2) 1982 $9.95

Wiley paperback guides teach data file programming and game programming on the Apple® and a wide range of other practical and recreational techniques. They've already helped more than a million people learn how to program, use, and enjoy microcomputers. Look for them all at your favorite bookshop or computer store.

1807 1982

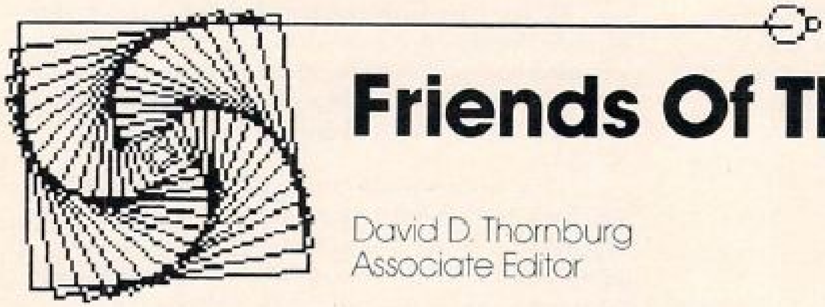
JOHN WILEY & SONS, Inc.
605 Third Avenue
New York, N. Y. 10158
In Canada: 22 Worcester Road, Rexdale, Ontario M9W 1L1 3-6817

Apple®, Apple II®, and Applesoft® are registered trademarks of Apple Computer, Inc.

```
7902  A9 88            LDA #<V5         ;RESULT AND
7904  20 73 D7   CM4   JSR FADD         ;ADD IT

7907                  ;10. SAVE RESULT IN SECOND OUTPUT VARIABLE

7907  20 19 79   TERM  JSR DEST         ;LOOK UP,SAVE RESULT

790A                  ;11. FIX UP THE STACK AND CHRGET ADDRESS

790A  68               PLA
790B  85 78            STA TXTPTR+1
790D  68               PLA
790E  85 77            STA TXTPTR
7910  60               RTS

7911  4C 03 CE   ERR   JMP STXERR       ;PRINT 'SYNTAX' & EXIT
7914  A2 A3      TYPMIS LDX #$A3        ;PRINT 'TYPE MISMATCH'
7916  4C 57 C3         JMP PRTERR       ;AND EXIT

7919                  ;LOOK UP DESTINATION AND STORE FACC THERE

7919  AD 70 79   DEST  LDA OPCHAR       ;SEE IF MULTIPLY
791C  C9 AC            CMP #172         ;TOKEN FOr '*'
791E  F0 10            BEQ D1           ;SKIP NORMLZ.IF MULTIPLY
7920  20 18 DB         JSR FACALT       ;PUT FACC INTO AFAC
7923  A5 63            LDA FSIGN        ;SAVE FACC SIGN
7925  48               PHA
7926  A0 79            LDY #>V6         ;IF DIVIDE THEN
7928  A9 8D            LDA #<V6         ;NORMLZ BY MAG SQUARED
792A  20 11 DA         JSR FDIV1        ;OF DIVISOR
792D  68               PLA              ;RESTORE FACC SIGN
792D  68               PLA              ;RESTORE FACC SIGN
792E  85 63            STA FSIGN
7930  20 70 00   D1    JSR CHRGET       ;MOVE PAST COMMA
7933  A0 05            LDY #5           ;SAVE FACC IN CASE OF
7935  B9 5E 00   D1A   LDA FACC,Y       ;SUBSCRIPTED VARIABLES
7938  48               PHA
7939  88               DEY
793A  10 F9            BPL D1A
793C  20 6D CF         JSR LOOKUP       ;GET DESTINATION ADDR
793F  A0 00            LDY #0           ;RESTORE THE FACC
7941  A2 05            LDX #5
7943  68         D1B   PLA
7944  99 5E 00         STA FACC,Y
7947  C8               INY
7948  CA               DEX
7949  10 F8            BPL D1B
794B  A5 07            LDA STRFLG       ;CHECK FOR STRING TYPE
794D  D0 C5            BNE TYPMIS       ;AND BRANCH IF IT IS
794F  A5 08            LDA INTFLG       ;CHECK FOR INTEGER
7951  F0 0E            BEQ D2

7953                  ;CONVERT TO INTEGER FORMAT IF THE
7953                  ;DESTINATION VARIABLES IS INTEGER

7953  20 9A D0         JSR FLPINT       ;CONVERT RESULT
7956  A0 01            LDY #1
7958  B9 61 00   D3    LDA FACC+3,Y     ;TRANSFER 2 BYTES
795B  91 44            STA (VARADR),Y   ;FROM FACC TO MEMORY
795D  88               DEY
795E  10 F8            BPL D3
7960  60               RTS
7961  A4 45      D2    LDY VARADR+1     ;FETCH
7963  A6 44            LDX VARADR       ;ADDRESS
7965  20 E0 DA         JSR STFACC       ;AND SAVE RESULT
7968  60               RTS
7969                  ;SAVE FACC INTO TEMPORARY V5

7969  A0 79      SAVTMP LDY #>V5        ;SET UP V5 ADDRESS
796B  A2 88            LDX #<V5         ;FOR TRANSFER
796D  4C E0 DA         JMP STFACC

7970                  ;STORAGE FOR VARIABLES AND CONSTANTS

7970                  OPCHAR *=*+1              ;OPERATION CHAR. + - * /
7971                  PARMS  *=*+1              ;# OF PARAMETERS TO GO
7972                  ASAVE  *=*+2              ;LINE SCAN ADDRESS
7974                  V1     *=*+5              ;FIRST ARGUMENT
7979                  V2     *=*+5              ;SECOND ARGUMENT
797E                  V3     *=*+5              ;THIRD ARGUMENT
7983                  V4     *=*+5              ;FOURTH ARGUMENT
7988                  V5     *=*+5              ;TEMPORARY REGISTER #1
798D                  V6     *=*+5              ;TEMPORARY REGISTER #2

7992                  ;VARIABLE TABLE

7992  88 79           VTAB   .WORD V5,V4,V3,V2,V1,V6
7994  83 79
7996  7E 79
7998  79 79
799A  74 79
799C  8D 79
799E                         .END
```

©

COMPUTE!
The Resource.



A Monthly Column

Friends Of The Turtle

David D. Thornburg
Associate Editor

Recursion – Part 2

Last time, we explored recursion as a powerful programming tool. The basic elements of a recursive procedure include:

1. A conditional statement to tell when to stop the recursive process;
2. A series of commands to be executed at each recursive level; and,
3. The use of the procedure itself with, perhaps, new values for the procedure's variables.

The sequence and intermixing of these elements determine the type of recursive process being followed. Recursion can range from simple looping to the more complex forms we used for drawing fractals.

Because of the obvious visual relationship between certain fractals and the recursive procedures that generate them, we will examine some more of these this month.

Before doing that, however, let's make a small digression to examine the difference between the conditional branching commands commonly used with Logo programs for the Apple computer and the conditional branching command used by TI Logo.

The structure of the command we have been using is:

```
IF predicate instructionlist
```

This means that the structure of the command is the word IF followed by an operation whose result is either true or false (the predicate), followed by a list of instructions to be executed if the predicate is true. An alternate form of this command is:

```
IF predicate THEN instructionlist
```

This form of the command is common to most BASICs as well, and might be familiar to many of you.

TI Logo uses a different type of conditional command, one which is more reminiscent of PILOT. In TI Logo the IF ... THEN ... construction is replaced by:

```
TEST predicate
IFT instructionlist1
```

and also

```
IFF instructionlist2
```

This construction allows you to test a predicate in a line all by itself, and to then execute certain instructions selectively, based on the result of the test, anywhere after the TEST command. The command IFT will execute instructionlist if the result of the test was true, and the command IFF will execute the list if the result was false.

In Apple Logo our conditional command in the fractal procedure is:

```
IF :SIZE < :LIMIT [FORWARD :SIZE STOP]
```

In TI LOGO this would be replaced by:

```
TEST :SIZE < :LIMIT
IFT FORWARD :SIZE STOP
```

One other note for TI Logo users: you may find that your turtle's pen "runs out of ink" on the more complex curves. You might want to try drawing smaller versions of them to minimize this problem. Of course, you should be sure to clear the screen before drawing anything, just to be sure you have recovered as much "ink" as possible.

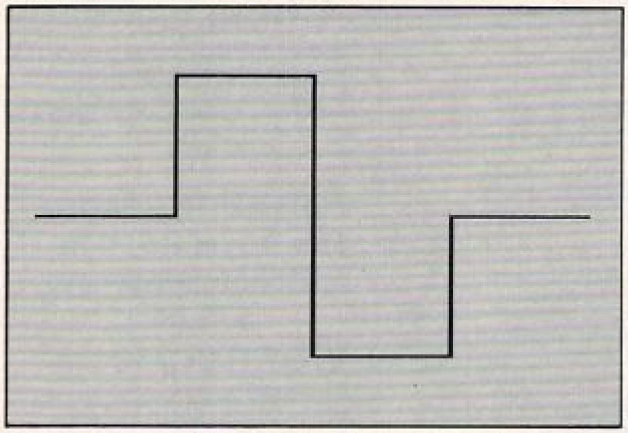
And now, on with the show!

One type of fractal that generates pretty pictures is the Koch curve we drew last time. In its most general form, we can define the motif for this type of curve as starting with a horizontal line, making some construction using line segments of the same length, and ending with a horizontal line on the same level as the first one. The following three fractals are particularly pleasing to me and to the people who have seen them exhibited at shows, so I am pleased to also share them with you. As in the past, all procedures will be shown in Apple Logo, and you can easily translate these to any other version of the language you might be using.

Before creating the curves, we will define a general setup procedure that puts the turtle in the correct starting position and orientation for each curve:

```
TO SETUP :LIST
PENUP
SETPOS :LIST
SETHEADING 90
PENDOWN
END
```

The first curve we will explore is a square meander.



The procedure for creating fractals based on this figure is the following:

```
TO MEANDER :SIZE :LIMIT
IF :SIZE < :LIMIT [FORWARD :SIZE STOP]
MEANDER :SIZE / 4 :LIMIT
LEFT 90
MEANDER :SIZE / 4 :LIMIT
RIGHT 90
MEANDER :SIZE / 4 :LIMIT
RIGHT 90
REPEAT 2 [MEANDER :SIZE / 4 :LIMIT]
LEFT 90
MEANDER :SIZE / 4 :LIMIT
LEFT 90
MEANDER :SIZE / 4 :LIMIT
RIGHT 90
MEANDER :SIZE / 4 :LIMIT
END
```

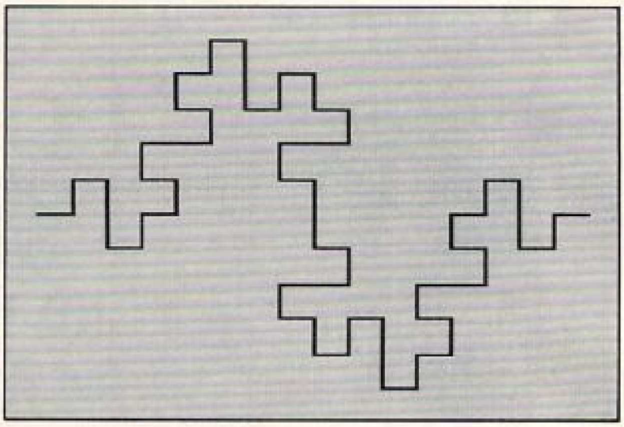
Before using this procedure, let's examine it. The first thing to notice is that the value of SIZE is reduced by a factor of four for each successive use of the procedure. The reason for this is that the total horizontal extent of the original motif is four times the length of the line segment. The second thing to notice is that the double length of line in the motif is created by a double repetition of the procedure. To see the motif, enter:

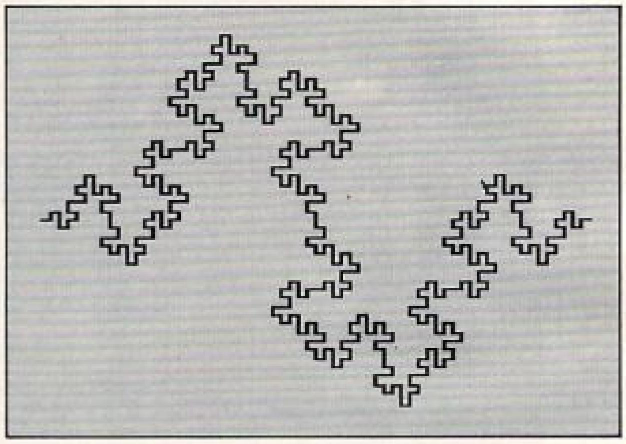
```
CLEARSCREEN
SETUP [-128 0]
MEANDER 256 256
```

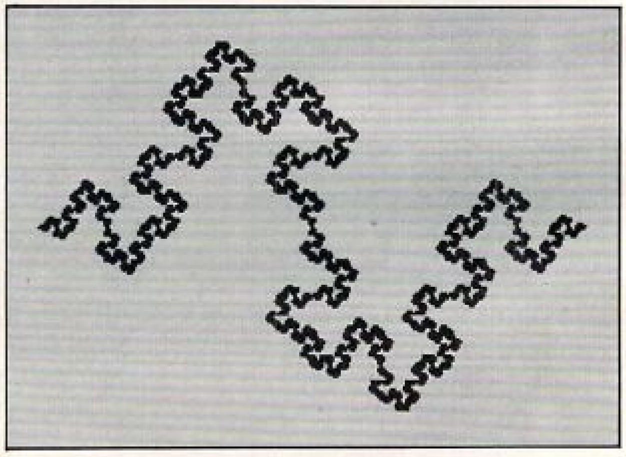
Successive generations can be seen by entering:

```
MEANDER 256 64
MEANDER 256 16
MEANDER 256 4
```

(Remember to clear the screen and use the SETUP procedure before drawing each curve.)

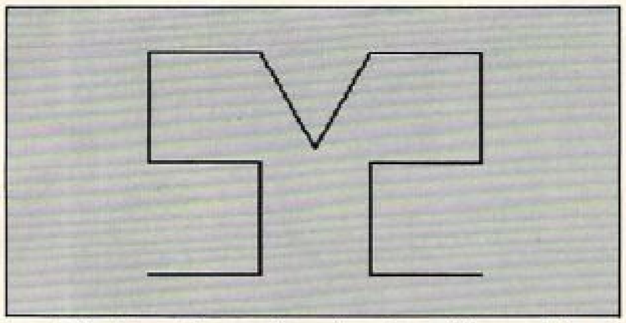






As you look at each successive generation of this figure, it is interesting to note the development of secondary meanders resulting in a final highly convoluted (but strangely symmetrical) form.

The second curve I want to share is called the T-shirt fractal, since it was designed for use on a T-shirt (write me at Friends of the Turtle for details). In making this design, I thought that a fractal T-shirt should use a T-shirt fractal, thus carrying the recursive process one step backwards to the overall shirt itself. The motif I designed looks like this:



The fractal procedure based on this motif is given by:

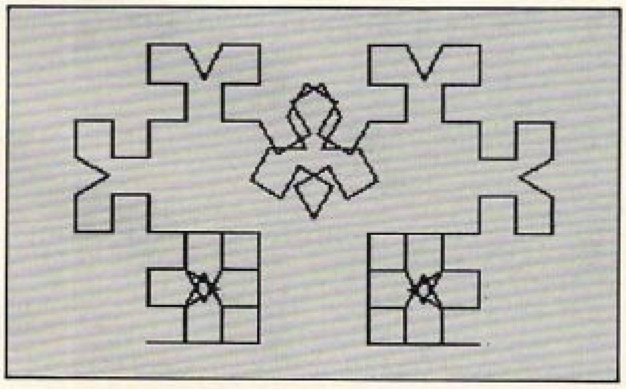
```
TO TSHIRT :SIZE :LIMIT
IF :SIZE < :LIMIT [FORWARD :SIZE STOP]
TSHIRT :SIZE / 3 :LIMIT
LEFT 90
TSHIRT :SIZE / 3 :LIMIT
LEFT 90
TSHIRT :SIZE / 3 :LIMIT
RIGHT 90
TSHIRT :SIZE / 3 :LIMIT
RIGHT 90
TSHIRT :SIZE / 3 :LIMIT
RIGHT 60
TSHIRT :SIZE / 3 :LIMIT
LEFT 120
TSHIRT :SIZE / 3 :LIMIT
RIGHT 60
TSHIRT :SIZE / 3 :LIMIT
RIGHT 90
TSHIRT :SIZE / 3 :LIMIT
RIGHT 90
TSHIRT :SIZE / 3 :LIMIT
LEFT 90
TSHIRT :SIZE / 3 :LIMIT
LEFT 90
TSHIRT :SIZE / 3 :LIMIT
END
```

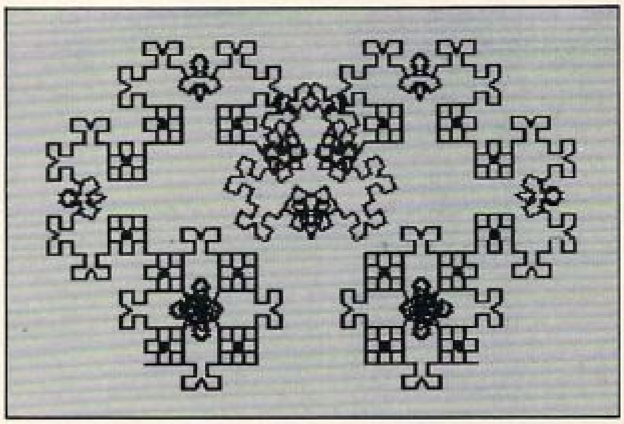
To generate the motif on the display, enter:

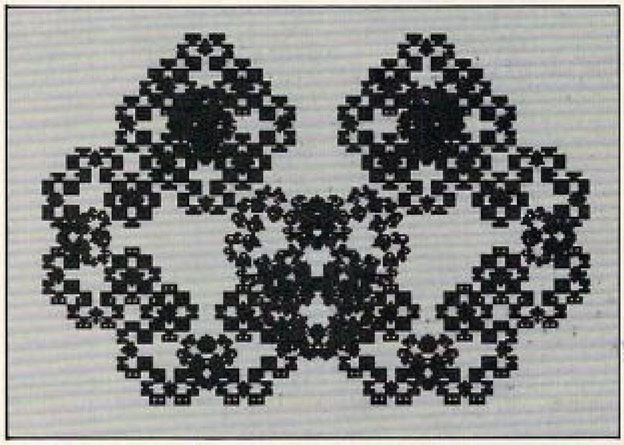
```
CLEARSCREEN
SETUP [-81 -60]
TSHIRT 162 162
```

Successive generations can be formed with the following commands:

```
TSHIRT 162 54
TSHIRT 162 18
```



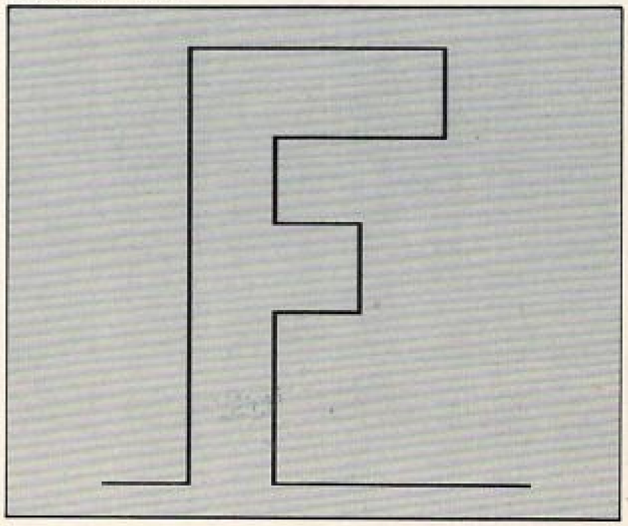




TSHIRT 162 6

Notice that, for this pattern, there is a lot of overlapping in successive generations that makes it harder to identify the original motif. But, if you look closely, you will be able to see the motif hidden (in full size) in each generation.

The last pattern I wanted to show is from a piece of artwork entitled *F is for Fractal.* The motif is quite simple:



The procedure for this curve is a bit on the lengthy side:

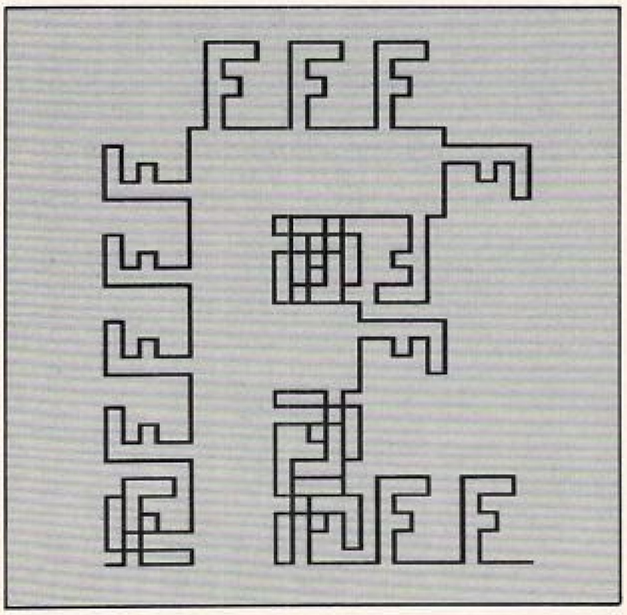
```
TO F :SIZE :LIMIT
IF :SIZE < :LIMIT [FORWARD :SIZE STOP]
F :SIZE / 5 :LIMIT
LEFT 90
REPEAT 5 [F :SIZE / 5 :LIMIT]
RIGHT 90
REPEAT 3 [F :SIZE / 5 :LIMIT]
RIGHT 90
F :SIZE / 5 :LIMIT
RIGHT 90
REPEAT 2 [F :SIZE / 5 :LIMIT]
LEFT 90
F :SIZE / 5 :LIMIT
LEFT 90
F :SIZE / 5 :LIMIT
RIGHT 90
F :SIZE / 5 :LIMIT
RIGHT 90
F :SIZE / 5 :LIMIT
LEFT 90
REPEAT 2 [IF :SIZE / 5 :LIMIT]
LEFT 90
REPEAT 3 [F :SIZE / 5 :LIMIT]
END
```

The motif can be generated by entering:

```
CLEARSCREEN
SETUP [-85 -110]
F 175 175
```

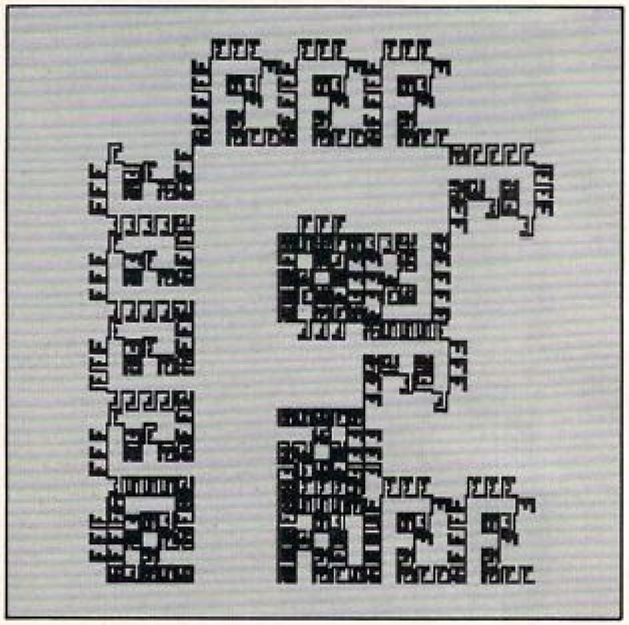
Further generations are created with the commands:

```
F 175 35
F 175 7
```



What I find particularly interesting is the manner in which the figure of the F in the motif becomes the background in the third generation.

By now, you probably have recursive programming firmly under control. You should con-



tinue to experiment on your own. The results may surprise you with their beauty!

Calling All Atari PILOTs

COMPUTE! reader Elliot Maggin sent me a delightful extension of a fractal program we described some months back. His program generates King Tut's Headdress. I think you will like the result.

```
2 R:*****************
3 R:*               *
4 R:*   90-DEGREE   *
5 R:*               *
6 R:*    FRACTAL    *
7 R:*               *
8 R:*****************
10 GR:PEN RED
20 GR:CLEAR
30 C:#A=54
40 GR:GOTO -79,-31
50 GR:TURNTO 90
60 U:*F0
70 GR:PEN BLUE
80 GR:GOTO -24,-32;TURN -90;FILL #A
90 GR:PEN RED
100 C:#A=#A/3
110 GR:GOTO -79,-31
120 GR:TURNTO 90
130 U:*F1
140 C:#A=#A/3
150 GR:GOTO -79,-31
160 GR:TURNTO 90
170 U:*F2
180 C:#A=#A/3
190 GR:GOTO -79,-33
200 GR:TURNTO 90
```

```
210 GR:PEN YELLOW
220 U:*F3
230 T:          KING TUT'S HEADDRESS
240 E:
250 *FO
260 GR:DRAW #A
270 GR:TURN -90
280 GR:DRAW #A
290 GR:TURN 90
300 GR:DRAW #A
310 GR:TURN 90
320 GR:DRAW #A
330 GR:TURN -90
340 GR:DRAW #A
350 E:
360 *F1
370 U:*FO
380 GR:TURN -90
390 U:*FO
400 GR:TURN 90
410 U:*FO
420 GR:TURN 90
430 U:*FO
440 GR:TURN -90
450 U:*FO
460 E:
470 *F2
480 U:*F1
490 GR:TURN -90
500 U:*F1
510 GR:TURN 90
520 U:*F1
530 GR:TURN 90
540 U:*F1
550 GR:TURN -90
560 U:*F1
570 E:
580 *F3
590 U:*F2
600 GR:TURN -90
610 U:*F2
620 GR:TURN 90
630 U:*F2
640 GR:TURN 90
650 U:*F2
660 GR:TURN -90
670 U:*F2
680 E:
```

A Year-end Note To All

Before leaving this year behind, I thought you should know some of the things we have in store for you in 1983. First, I have received the Turtle Graphics package for the VIC designed and manufactured by HES, and will report on it in January. Also, I am now using the Radio Shack Color Logo package and will be reporting on it in the same issue. Those of you who are interested in fractals may be interested in *The Fractal Geometry of Nature*, a new book by the father of this study, Benoit Mandelbrot. I will be reviewing this book and commenting on the controversy in this field in a forthcoming "Computers and Society" column.

In the meantime, let me know what *you* want to read, and I'll see what I can do to meet your needs.

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

©

SOFTWARE AND ACCESSORIES FOR COMMODORE'S VIC-20*

Examples:
Super Checkbook Balancing $8.95
Multiplication Tables, Practice/test $7.95
Numeric Keypad, hardware/software $39.95
Cognitive Psychology Programs (each) $19.95
(Prices include postage)

Send SASE for info on software for the home, school, and psychology laboratory.

SCIENTIFIC AND EDUCATIONAL SOFTWARE, INC.
P.O. BOX 54 DAYTON, OHIO 45420

*VIC-20 is a registered trademark of Commodore Business Machines.

Micro Power Bench™

- Single Switch Control of CPU and Peripherals
- Built in circuit breaker protects your system
- Four power expansion outlets
- Choice of high quality oak or walnut
- Compatible with IBM, APPLE, TRS-80 and others

SAVE! SAVE! ONLY $79.00

Order Direct
800-343-4311
Master Charge and Visa Accepted

CAB-TEK, Inc.
Riverside St., Nashua NH 03062
CIVILIZING COMPUTERS

MX 80
ACOUSTIC ENCLOSURE $99!

DEALER INQUIRIES INVITED

For Apple Logo and Atari PILOT, this program provides a way to make the turtle draw the numerals from zero to nine. Using the techniques shown, you will be able to extend this method to include the alphabet as well. TI and Radio Shack Logo users can build a program from the examples given.

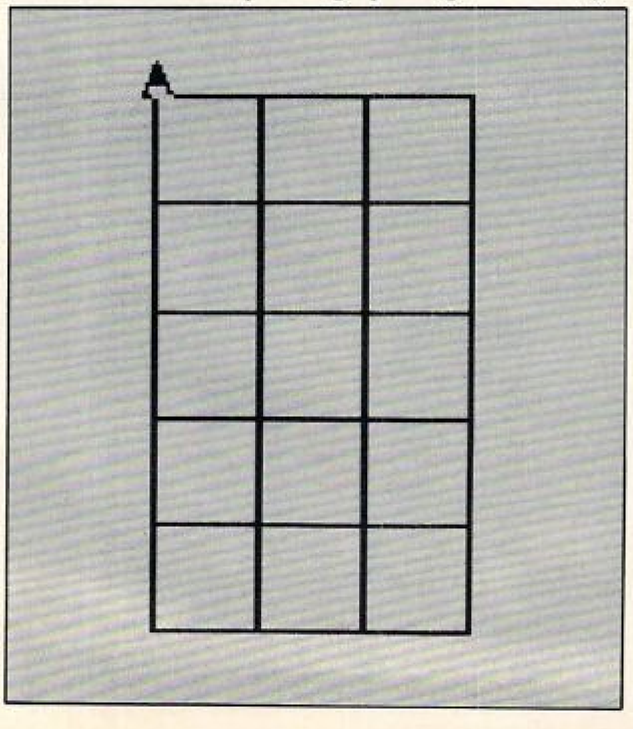
Making The Turtle Count

David D. Thornburg
Associate Editor

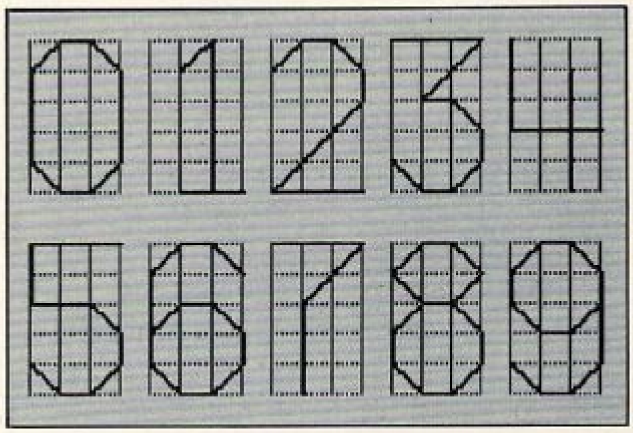
With the single exception of Apple SuperPILOT, none of the popular turtle graphics systems with which I am familiar allows the user to freely intermix text and graphics. One solution to this problem is to teach the turtle how to write!

If we are going to have the turtle draw numbers on the screen, we should pick a number drawing technique that lets us draw numbers of any size, orientation, location, and color we choose. The result will be a text display system that is more powerful than traditional dot matrix characters.

The character field I have chosen is three units wide and five units high. If the resultant characters are too high and skinny on your display, you will want to modify our method slightly to satisfy your own taste. The turtle starts and ends each character at the upper left corner of the grid, with its orientation pointing up along the left edge.



Using this grid we can design the numerals we want to draw, as shown below:



Each procedure for drawing consists of picking the turtle's pen up, moving the turtle to the starting position, putting the pen down, drawing the character in one continuous motion, picking the pen up, and moving the turtle back to its starting position and orientation. The shapes of the characters are defined so that each line segment is either along a grid length or along a grid diagonal. Since the length of the diagonal is larger than the grid length by the square root of two, our procedures need to incorporate this number.

This is fairly easy for the Apple Logos since they all use floating point arithmetic. Atari PILOT, TI Logo, and Radio Shack Color Logo, however, use only integer arithmetic. So, for these languages, we need to find a way to approximate the multiplication of a number by the square root of two. Obviously, we can't use the decimal number 1.414 because the language won't know what to do with it. Similarly, we can't just multiply by (1414/1000) because, if this division is performed first, the result will be one! But, if we first multiply the grid size by 1414 and then do the division by 1000, the result should be an effective approximation.

The following listings for the ten numeral procedures are shown in Apple Logo and Atari PILOT. Users of TI Logo, Radio Shack Color Logo, and other languages using integer arithmetic will have to mix and match from these two sets of procedures as needed.

Apple LOGO

```
TO ZERO :SIZE
MAKE "ROOT :SIZE * 1.41421
PENUP
BACK :SIZE
PENDOWN
RIGHT 45 FORWARD :ROOT
RIGHT 45 FORWARD :SIZE
RIGHT 45 FORWARD :ROOT
RIGHT 45 FORWARD :SIZE * 3
RIGHT 45 FORWARD :ROOT
RIGHT 45 FORWARD :SIZE
RIGHT 45 FORWARD :ROOT
RIGHT 45 FORWARD :SIZE * 3
PENUP
FORWARD :SIZE
PENDOWN
END

TO ONE :SIZE
MAKE "ROOT :SIZE * 1.41421
PENUP
BACK :SIZE RIGHT 90
FORWARD :SIZE LEFT 45
PENDOWN
FORWARD :ROOT
RIGHT 135 FORWARD :SIZE * 5
RIGHT 90 FORWARD :SIZE
BACK :SIZE * 2
PENUP
RIGHT 90 FORWARD :SIZE * 5
LEFT 90 FORWARD :SIZE * 3
RIGHT 90
PENDOWN
END

TO TWO :SIZE
MAKE "ROOT :SIZE * 1.41421
PENUP
BACK :SIZE
PENDOWN
RIGHT 45 FORWARD :ROOT
RIGHT 45 FORWARD :SIZE
RIGHT 45 FORWARD :ROOT
RIGHT 45 FORWARD :SIZE
RIGHT 45 FORWARD :ROOT * 3
LEFT 135 FORWARD :SIZE * 3
PENUP
LEFT 90 FORWARD :SIZE * 5
LEFT 90 FORWARD :SIZE * 3
RIGHT 90
PENDOWN
END

TO THREE :SIZE
MAKE "ROOT :SIZE * 1.41421
RIGHT 90 FORWARD :SIZE * 3
RIGHT 135 FORWARD :ROOT * 2
LEFT 135 FORWARD :SIZE
RIGHT 45 FORWARD :ROOT
RIGHT 45 FORWARD :SIZE
RIGHT 45 FORWARD :ROOT
RIGHT 45 FORWARD :SIZE
RIGHT 45 FORWARD :ROOT
PENUP
RIGHT 45 FORWARD :SIZE * 4
PENDOWN
END

TO FOUR :SIZE
MAKE "ROOT :SIZE * 1.41421
RIGHT 180 FORWARD :SIZE * 3
LEFT 90 FORWARD :SIZE * 3
BACK :SIZE
LEFT 90 FORWARD :SIZE * 2
BACK :SIZE * 4
PENUP
FORWARD :SIZE * 5 LEFT 90
FORWARD :SIZE * 2 RIGHT 90
PENDOWN
END

TO FIVE :SIZE
MAKE "ROOT :SIZE * 1.41421
RIGHT 90 FORWARD :SIZE * 3
BACK :SIZE * 3
RIGHT 90 FORWARD :SIZE * 2
LEFT 90 FORWARD :SIZE * 2
RIGHT 45 FORWARD :ROOT
RIGHT 45 FORWARD :SIZE
RIGHT 45 FORWARD :ROOT
RIGHT 45 FORWARD :SIZE
RIGHT 45 FORWARD :ROOT
PENUP
RIGHT 45 FORWARD :SIZE * 4
PENDOWN
END

TO SIX :SIZE
MAKE "ROOT :SIZE * 1.41421
PENUP
RIGHT 90 FORWARD :SIZE * 3
RIGHT 90 FORWARD :SIZE
RIGHT 135
PENDOWN
FORWARD :ROOT
LEFT 45 FORWARD :SIZE
LEFT 45 FORWARD :ROOT
LEFT 45 FORWARD :SIZE * 3
LEFT 45 FORWARD :ROOT
LEFT 45 FORWARD :SIZE
LEFT 45 FORWARD :ROOT
LEFT 45 FORWARD :SIZE
LEFT 45 FORWARD :ROOT
LEFT 45 FORWARD :SIZE
LEFT 45 FORWARD :ROOT
PENUP
RIGHT 135 FORWARD :SIZE * 3
PENDOWN
END

TO SEVEN :SIZE
MAKE "ROOT :SIZE * 1.41421
RIGHT 90 FORWARD :SIZE * 3
RIGHT 135 FORWARD :ROOT * 2
LEFT 45 FORWARD :SIZE * 3
PENUP
RIGHT 180 FORWARD :SIZE * 5
LEFT 90 FORWARD :SIZE
RIGHT 90
PENDOWN
END

TO EIGHT :SIZE
MAKE "ROOT :SIZE * 1.41421
PENUP
RIGHT 90 FORWARD :SIZE
PENDOWN
FORWARD :SIZE
RIGHT 45 FORWARD :ROOT
RIGHT 90 FORWARD :ROOT
RIGHT 45 FORWARD :SIZE
LEFT 45 FORWARD :ROOT
LEFT 45 FORWARD :SIZE
LEFT 45 FORWARD :ROOT
LEFT 45 FORWARD :SIZE
LEFT 45 FORWARD :ROOT
LEFT 45 FORWARD :SIZE
LEFT 45 FORWARD :ROOT
LEFT 45 FORWARD :SIZE
RIGHT 45 FORWARD :ROOT
RIGHT 90 FORWARD :ROOT
PENUP
LEFT 135 FORWARD :SIZE
RIGHT 90
PENDOWN
END

TO NINE :SIZE
MAKE "ROOT :SIZE * 1.41421
PENUP
RIGHT 90 FORWARD :SIZE * 3
```

Atari PILOT

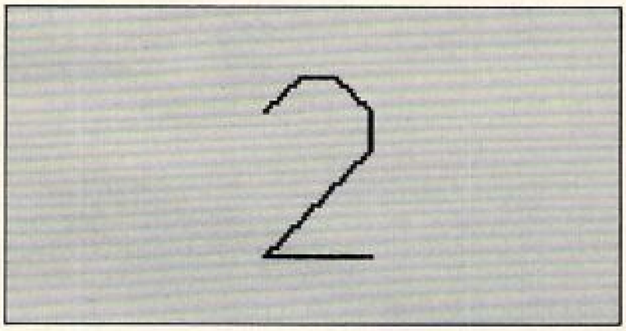
```
*ZERO
C: #R=(#S*1414)/1000
GR: PENUP
GR: DRAW -#S
GR: PEN YELLOW
GR: TURN 45; DRAW #R
GR: TURN 45; DRAW #S
GR: TURN 45; DRAW #R
GR: TURN 45; DRAW #S*3
GR: TURN 45: DRAW #R
GR: TURN 45: DRAW #S
GR: TURN 45: DRAW #R
GR: TURN 45; DRAW #S*3
GR: PENUP
GR: DRAW #S
GR: PEN YELLOW
E:

*ONE
C: #R=(#S*1414)/1000
GR: PENUP
GR: DRAW -#S; TURN 90
GR: DRAW #S; TURN -45
GR: PEN YELLOW
GR: DRAW #R
GR: TURN 135; DRAW #S*5
GR: TURN 90; DRAW #S
GR: DRAW -#S*2
GR: PENUP
GR: TURN 90; DRAW #S*5
GR: TURN -90; DRAW #S*3
GR: TURN 90
GR: PEN YELLOW
E:

*TWO
C: #R=(#S*1414)/1000
GR: PENUP
GR: DRAW -#S
GR: PEN YELLOW
GR: TURN 45; DRAW #R
GR: TURN 45; DRAW #S
GR: TURN 45; DRAW #R
GR: TURN 45; DRAW #S
GR: TURN 45; DRAW #R*3
GR: TURN -135; DRAW #S*3
GR: PENUP
GR: TURN -90; DRAW #S*5
GR: TURN -90; DRAW #S*3
GR: RIGHT 90
GR: PEN YELLOW
E:

*THREE
C: #R=(#S*1414)/1000
GR: TURN 90; DRAW #S*3
GR: TURN 135; DRAW #R*2
GR: TURN -135; DRAW #S
GR: TURN 45; DRAW #R
GR: TURN 45; DRAW #S
GR: TURN 45; DRAW #R
GR: TURN 45; DRAW #S
GR: TURN 45; DRAW #R
GR: PENUP
GR: TURN 45; DRAW #S*4
GR: PEN YELLOW
E:

*FOUR
C: #R=(#S*1414)/1000
GR: TURN 180; DRAW #S*3
GR: TURN -90; DRAW #S*3
GR: DRAW -#S
GR: TURN -90; DRAW #S*2
GR: DRAW -#S*4
GR: PENUP
GR: DRAW #S*5; TURN -90
GR: DRAW #S*2; TURN 90
GR: PEN YELLOW
E:

*FIVE
C: #R=(#S*1414)/1000
GR: TURN 90; DRAW #S*3
GR: DRAW -#S*3
GR: TURN 90; DRAW #S*2
GR: TURN -90; DRAW #S*2
GR: TURN 45; DRAW #R
GR: TURN 45; DRAW #S
GR: TURN 45; DRAW #R
GR: TURN 45; DRAW #S
GR: TURN 45; DRAW #R
GR: PENUP
GR: TURN 45; DRAW #S*4
GR: PEN YELLOW
E:

*SIX
C: #R=(#S*1414)/1000
GR: PENUP
GR: TURN 90; DRAW #S*3
GR: TURN 90; DRAW #S
GR: TURN 135
GR: PEN YELLOW
GR: DRAW #R
GR: TURN -45; DRAW #S
GR: TURN -45; DRAW #R
GR: TURN -45; DRAW #S*3
GR: TURN -45; DRAW #R
GR: TURN -45; DRAW #S
GR: TURN -45; DRAW #R
GR: TURN -45; DRAW #S
GR: TURN -45; DRAW #R
GR: TURN -45; DRAW #S
GR: TURN -45; DRAW #R
GR: PENUP
GR: TURN 135; DRAW #S*3
GR: PEN YELLOW
E:

*SEVEN
C: #R=(#S*1414)/1000
GR: TURN 90; DRAW #S*3
GR: TURN 135; DRAW #R*2
GR: TURN -90; DRAW #S*3
GR: PENUP
GR: TURN 180; DRAW #S*5
GR: TURN -90; DRAW #S
GR: TURN 90
GR: PEN YELLOW
E:

*EIGHT
C: #R=(#S*1414)/1000
GR: PENUP
GR: TURN 90; DRAW #S
GR: PEN YELLOW
GR: DRAW #S
GR: TURN 45; DRAW #R
GR: TURN 90; DRAW #R
GR: TURN 45; DRAW #S
GR: TURN -45; DRAW #R
GR: TURN -45; DRAW #S
GR: TURN -45; DRAW #R
GR: TURN -45; DRAW #S
GR: TURN -45; DRAW #R
GR: TURN -45; DRAW #S
GR: TURN -45; DRAW #R
GR: TURN -45; DRAW #S
GR: TURN 45; DRAW #R
GR: TURN 90; DRAW #R
GR: PENUP
GR: TURN -135; DRAW #S
GR: RIGHT 90
GR: PEN YELLOW
E:

*NINE
C: #R=(#S*1414)/1000
GR: PENUP
GR: TURN 90; DRAW #S*3
```

```
RIGHT 90 FORWARD :SIZE          | GR: TURN 90; DRAW #S
RIGHT 135                       | GR: TURN 135
PENDOWN                         | GR: PEN YELLOW
FORWARD :ROOT                   | GR: DRAW #R
LEFT 45 FORWARD :SIZE           | GR: TURN -45; DRAW #S
LEFT 45 FORWARD :ROOT           | GR: TURN -45; DRAW #R
LEFT 45 FORWARD :SIZE           | GR: TURN -45; DRAW #S
LEFT 45 FORWARD :ROOT           | GR: TURN -45; DRAW #R
LEFT 45 FORWARD :SIZE           | GR: TURN -45; DRAW #S
LEFT 45 FORWARD :ROOT           | GR: TURN -45; DRAW #R
LEFT 45 FORWARD :SIZE           | GR: TURN -45; DRAW #S
RIGHT 180 FORWARD :SIZE * 3     | GR: TURN 180; DRAW #S*3
RIGHT 45 FORWARD :ROOT          | GR: TURN 45; DRAW #R
RIGHT 45 FORWARD :SIZE          | GR: TURN 45; DRAW #S
RIGHT 45 FORWARD :ROOT          | GR: TURN 45; DRAW #R
PENUP                           | GR: PENUP
RIGHT 45 FORWARD :SIZE * 4      | GR: RIGHT 45; DRAW #S*4
PENDOWN                         | GR: PEN YELLOW
END                             | E:
```

Now that these characters have been defined, it is easy to place a numeral anywhere you want on the graphics screen. For example, if (in LOGO) you enter:

CLEARSCREEN
HIDETURTLE
TWO 10

you will see the numeral 2 on the screen.



In Atari PILOT, the length of the grid unit is given by #S, so you must first enter:

C: #S=10
U: *TWO

to get this result.

But what about numbers longer than one digit? How does one print these? A LOGO procedure to print multiple digit numbers (using recursion) is shown below (you *have* been reading the "Friends of the Turtle" columns on recursion, haven't you?):

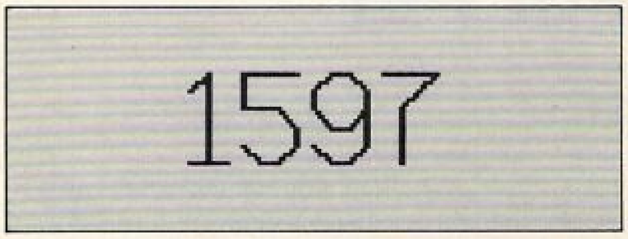
```
TO NUMB :LIST :SIZE
IF :LIST = [] [STOP]
RUN SENTENCE FIRST :LIST :SIZE
PENUP
RIGHT 90 FORWARD :SIZE * 4 LEFT 90
PENDOWN
NUMB BUTFIRST :LIST :SIZE
END
```

(Note: crafty Atari PILOT programmers will find at least two alternate ways to do this. At least one of these people will be kind enough to share the results with the rest of the readers.)

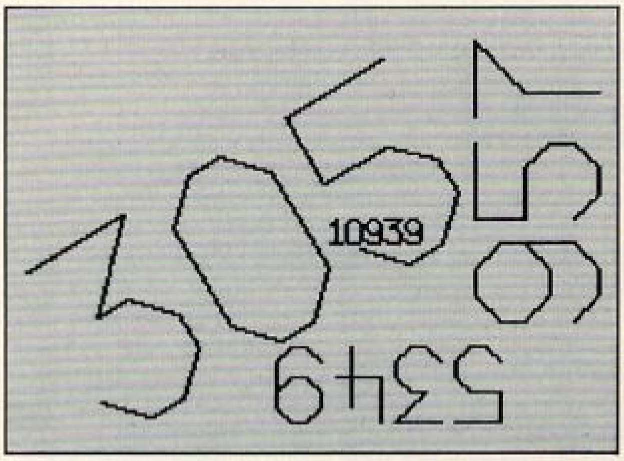
Now, with this procedure in hand, LOGO users should try entering something like:

CLEARSCREEN
NUMB [ONE FIVE NINE SEVEN] 5

to see what happens.



Experiment with different numbers, sizes, starting points, and orientations. You will find that you can print numbers at any angle. This is very handy for labeling graphs.

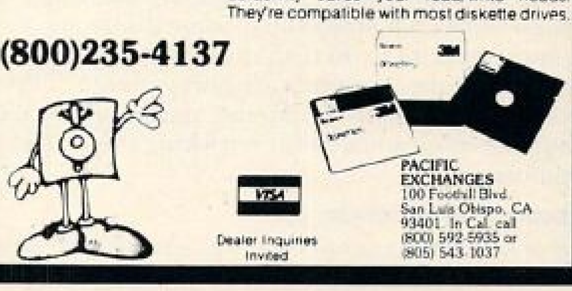


Expanding these ten numerals to the full alphabet is fairly straightforward. Any takers? ©

Scotch Diskettes

Rely on Scotch® diskettes to keep your valuable data safe. Dependable Scotch diskettes are tested and guaranteed error-free. The low abrasivity saves your read/write heads. They're compatible with most diskette drives.

(800)235-4137



Dealer Inquiries Invited

PACIFIC EXCHANGES
100 Foothill Blvd.
San Luis Obispo, CA
93401. In Cal. call
(800) 592-5935 or
(805) 543-1037

Review:

Apple Adventures

Dale Woolridge
Harrisburg, PA

Adventure games are older than Apple computers, and a high percentage of micro owners have played with them. These games give you a "world" containing dragons, demons, objects to be manipulated, etc. You use simple commands to move through the "world" and manipulate it.

Adventure – Colossal Cave

This is the original *Adventure* game, written first in FORTRAN for a PDP-10, by Willie Crowther and Don Woods. This program was implemented on the Apple by Master Jacobi. The program was compressed to fit entirely into 48K of RAM to avoid accesses to the disk.

Adventure has 15 treasures which add points to your score. It might not be obvious what a treasure is, so you might be tempted to pick up any object you find. There are 40 useful objects, but they have side effects. For example, the bird is afraid of the rod, and a certain magic word works only when you possess certain objects. The "world" is fairly large, containing 130 rooms. It is easy to find about a tenth of the rooms; the others are hard to find. In addition, there are 12 obstacles or opponents.

The game is complicated enough to keep you busy for a long time. If you are stumped, you can save the game to be resumed later. When you resume, you are asked if you want to load the saved game. If you say yes, you get back into the saved game, and the game is deleted from the disk. If you say no, you can start a new game while the saved game remains on the disk. You can save only one game.

Help, For A Price

A wizard, Arian, guides you through the world. A surprising, and amusing, feature of the game is that if you try many times to do a certain thing, but fail, the wizard will finally offer to help – for a price.

There is apparently a random element to the game. There is at least one situation in which you may or may not be killed, depending on chance.

The scoring scheme is somewhat unusual. You get points merely for discovering parts of the world and for finding objects. Getting killed costs you points. Your wizard might be able to bring you back to life, but you might lose the objects you were carrying.

The program is on a protected disk. The disk boots and the program loads in only nine seconds. At the beginning of the game a message appears briefly on the screen, and if you are a slow reader you might miss some of it. The message appears during the boot phase and disappears when the program executes. However, most of the program is well written and courteous to the user.

Adventureland

This Scott Adams' game has several features unusual in adventure games. The graphics were done using Penguin Software's Picture Editor, by Mark Pelczarski. The quality of the pictures is quite good. It takes 10-20 seconds, typically, to load a picture from the disk, and in case you don't have the time, the program lets you switch between graphics mode and all text mode. Often, a complete picture is "painted" on the screen, and then the disk drive comes on and certain objects are superimposed on the picture. This feature of the program gives you clues about the game, since the superimposed objects can generally move or be moved.

Use Peripherals

If you have a Votrax Type 'N Talk voice synthesizer, you can get the computer to speak the responses to your command. The responses will also be displayed on the screen.

If you have a lowercase adapter on your Apple, you can switch between all uppercase mode and upper/lowercase mode. And if you have a printer, you can get a hard copy of your adventure. The instruction booklet says that with some printer cards you might have to initialize the card in Applesoft before starting the adventure program. The Silentype printer does not require initialization before the game.

Another nice feature is that you can save up to four adventures to be resumed later. Considering that an adventure can occupy you for hours, this feature is desirable.

Before the game begins, you are invited to read an "open letter." The letter is a lecture on software piracy and includes several high resolution graphic pictures (of pirates, the American flag, etc.).

It is very important to have the proper mind-set when playing *Adventureland.* You must be able to tolerate some frustration, since you might get "stuck" in part of Adams' world. Also, you should realize that a game is not won in a few minutes of play; it might be complicated enough to keep you busy for weeks or months. Ideas may come to you while you are driving, and when you try them out that evening a whole new part of the world will be revealed to you.

The author's sense of humor is evident. He has apparently anticipated some of the commands you are likely to give and has prepared comebacks for you.

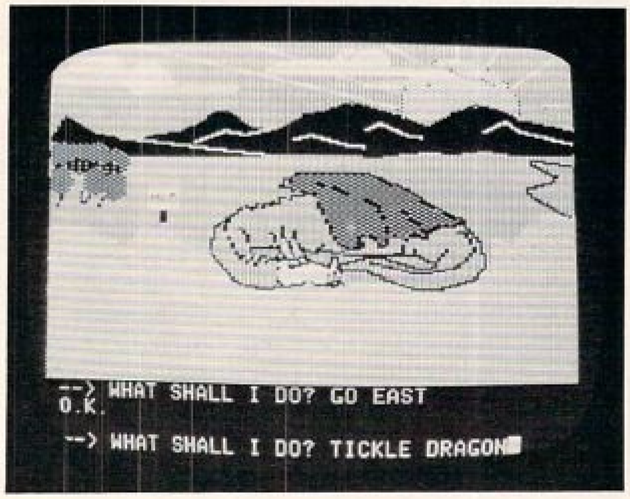
There is little randomness in *Adventureland.* As a rule, the same set of commands will have the same effects in different games. Success is obtained by using reason and common sense. However, there is an element of magic in the game; for example, you can come back to life if you give the right commands after being killed. There are also magic words.

It is very difficult to "crash" the program by giving bizarre input. It simply returns a message that it doesn't understand. Pressing RESET, however, will restart the game and clear out your adventure.

Adventure – Colossal Cave
Frontier Computing Inc.
P.O. Box 402
666 N. Main
Logan, UT 84321
$10 plus $1 shipping

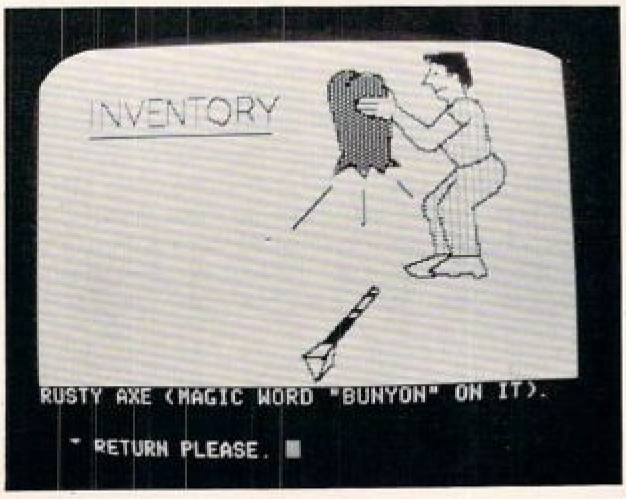
Adventureland
Adventure International
507 East Street
Box 3435
Longwood, FL 32750
$29.95 disk

©



```
--> WHAT SHALL I DO? GO EAST
O.K.
 --> WHAT SHALL I DO? TICKLE DRAGON
```

Teasing the dragon in Adventureland.



```
RUSTY AXE (MAGIC WORD "BUNYON" ON IT).

 " RETURN PLEASE.
```

Taking inventory in Adventureland.

VIC-20 OWNERS
Fed up with slow games?
The boredom is over!!
3 Fast games on one tape :
1. KILLER BEES
2. ALIEN DEATH RAY
3. VAULT OF DOOM
(Joystick and keyboard)
Full color & Sound effects
ALL 3 FOR ONLY $14.95
(check or money order)

MICRO-MANIA
P.O. BOX. 4110
ELKHART, IN
46514

A Monthly Column

COMPUTE! *welcomes Keith Falkner, whose "Extrapolations" column begins this month. Keith, who has extensive experience at all levels of computing, has contributed several excellent Apple articles to* **COMPUTE!** *in the past. To start his monthly column, he demonstrates how to use a simple BRUN to bring in the power of the renumber program – without affecting the program in memory. There's also a way to make yourself a simple assembler if you don't have the Mini-assembler.*

Extrapolations

Beat The "Applesoft Renumber" Blues

Keith Falkner, Toronto

On your System Master diskette there is a very powerful utility program called *Renumber*. This program can merge two Applesoft programs and can move several lines from one place to another within an Applesoft program. Of course, Renumber will also renumber the lines of an Applesoft pro-gram, and the options it offers in this function are as complete as anyone could wish.

Furthermore, Renumber is cleverly packaged as an Applesoft program so that no complicated machine language instructions are needed to run it.

Protecting Memory

When you run Renumber, a hidden machine language component relocates itself to the top 2048 bytes of memory, prevents Applesoft from overwriting it, and enables the ampersand (&) command. Thereafter you can LOAD, RUN, SAVE, etc., as usual, and the ampersand command invokes one of the three functions of Renumber. This is very clever packaging, because this way only one version of Renumber is needed for 32K or 48K Apples, regardless of the current upper limit of memory.

Setting MAXFILES or running the utility known as Program Line Editor both alter the upper limit of memory, but Renumber does not care. This versatility is commendable, but it comes at a price. If you have not bothered to run Renumber, but are working on an Applesoft program and wish to renumber it, you must first SAVE it, then run Renumber, then reload your Applesoft program. Generally, you do not need this flexibility. For example, if you have a 48K Apple, the machine language component eventually resides in locations $8E00-$95FF (36352-38399).

I'll show you how to save this machine language routine, together with a prologue to do the minimum initialization. Then a simple BRUN command will activate the essence of the Renumber program, without affecting any Applesoft program in memory. At the same time we will deal with the more or less well-known bug. If the program being renumbered contains a multiplication by a constant, such as X1 = J * 100, and there is a line number 100 which becomes, say, line number 80 upon renumbering, the constant may become 80 as well.

This is a consequence of the clever relocation routine which makes the machine language code function in whatever memory locations it occupies. Specifically, the token for LIST is replaced by the token for multiplication because the sequence of tokens $AC $B0 $BC is taken for the instruction LDY $BCB0, and the relocation routine changes this to LDY $CAB0.

So $BC, the token for LIST, has been replaced by $CA, the token for *. Hence, line number references following LIST (a rare verb to find in a BASIC program) can never be renumbered, and constants which appear to be line number references in a multiplication statement are subject to bogus renumbering! Fortunately, this is easy to fix.

One more thing should be done to Renumber. Some of us have a program to load PET tapes into our Apples, and some of these programs have spaces between the words or numbers in the program. In PETs this practice improves legibility, but not so in Apples, so Applesoft removes any extra spaces you may type in. Thus, Renumber does not expect spaces in, for example, GOSUB_______400. Those spaces prevent Renumber from changing that 400 if renumbering gives line 400 a new line number. The fix for this problem is included in Programs 1 and 2.

Now it's your turn to do some work: if you use DOS 3.2, type the lines in Program 1; if you use DOS 3.3, type the lines in Program 2. In either case, test your results as shown below.

Type in this trivial program:

```
1 INPUT X
2 IF X < 1 THEN 1
3 ON X GOSUB 39,87
27 END
39 LIST 87
45 RETURN
87 PRINT 99 * 39
99 GOTO 45
```

Now ready the renumbering routine:

BRUN BRENUMBER

Now renumber your program:

&
LIST

The result should look like this:

```
10 INPUT X
20 IF X < 1 THEN 10
30 ON X GOSUB 50,70
40 END
50 LIST 70
60 RETURN
70 PRINT 99 * 39
80 GOTO 60
```

With the stock Renumber program in a 48K Apple, line 50 would still say LIST 87 and line 70 would now say PRINT 99 * 50. Now type NEW ... the above is worthless. Don't proceed until you get it right, because an unreliable or inaccurate tool is much worse than none at all.

Here is what you have produced. *Brenumber* is a small (ten sector) binary program which loads into locations $8DE0-$95FF, sets the upper limit of memory to $8E00 (minus one), and sets up the ampersand (&) command to invoke the functions of Renumber. Brenumber may be used only in a 48K Apple, and then only when MAXFILES has its default value of three. There are no safeguards in Brenumber, so unpredictable results occur if these constraints aren't met.

Orderly Programs

Now, suppose you are working on an Applesoft program and you decide to renumber it. Without bothering to save it, just BRUN Brenumber and you have all the facilities of Renumber available. It's important to remember that the BRUN command *did not* renumber your program; it just enabled the ampersand (&) command which does the actual renumbering. So let's think of clever ways to use the Renumber program. The program, with 16 screens of instructions, can be formidable to try to understand, but it's worth learning.

Briefly, renumbering is done by typing the ampersand (&) and maybe some parameters. The parameters tell Renumber two things: what line numbers to assign and what portion of the program is to be renumbered. All the parameters are optional, the default being to renumber the whole program 10, 20, 30, etc.

FIRST = 1000 the first line number will be 1000.
INC = 20 successive line numbers increase by 20.
START = 5000 only lines 5000 and later will be renumbered.
END = 6990 only lines up to 6990 will be renumbered.

The FIRST and INC parameters are straightforward, so let's see how the START and END parameters can help us. One way I make my programs neat and readable, as well as accurate, is to have a main routine whose line numbers are less than 1000, and a menu which eventually says something like ON SEL GOSUB 1000,2000,3000, ... 11000, for example, if there are 11 selections from the main menu. Then I use line numbers 20000 and up for subordinate routines such as entitling the screen, formatting numbers, etc.

So how do I preserve this orderly scheme in renumbering the program? Well, consider the effect of these commands:

& F 100, S 0, E 999 (parameters can be abbreviated)
& F 1000, S 1000, E 1999
& F 2000, S 2000, E 2999

and so on, until finally

& F 20000, S 20000

The first command will renumber only the main routine; the second will renumber lines 1000-1999, etc., and the last will renumber only the elementary routines. All very fine, but who wants to type in 21 commands to renumber a program? Well, here is a simple six-line program to create an EXEC FILE named RENUM. Customize the program to suit yourself, then run it one time and keep its output on the same disk you have Brenumber on. Then, when you wish to renumber a program in the complex way outlined above, just type EXEC RENUM.

```
10 D$ = CHR$ (4):F$ = "RENUM":Q$ = CHR$ (34)
20 PRINT D$"OPEN"F$: PRINT D$"WRITE"F$: PRINT
    "MON I"
30 PRINT "IF PEEK(36352)<>164 THEN ?CHR$(4)"Q
    $"BRUN BRENUMBER"Q$
40 X = 100:Y = 999: FOR I = 0 TO 30: IF I THE
    N X = Y + 1:Y = Y + 1000
50 PRINT "& F"X",S"X",E"Y: NEXT
60 PRINT "? CHR$(7)": PRINT "NOMON I": PRINT
    D$"CLOSE"
```

RENUM can take several minutes to do its work on a large program, so you have an opportunity for a break. It is vital that you never press RESET while Renumber (or Brenumber) is operating – it's almost certain to destroy your program! The MON I statement at the start of the EXEC FILE causes each command to be listed as it is read from disk, so watch and wait patiently.

Hiding And Moving Lines

The HOLD and MERGE functions of the Renumber program are probably poorly understood; here is an example which barely hints at the power of these commands.

LOAD PHONE LIST	**from System Master disk**
BRUN BRENUMBER	**from the disk where you put it**
& 5400, F1000	**to make a gap for more lines of DATA**
SAVE PHONE INTERIM	**we need it on disk for a moment**
DEL 1,200	**discard the prologue and credits**
DEL 351,63999	**discard everything but DATA statements**
& F351, I1	**old DATA from 201-350 becomes 351-500**
& HOLD	**put 150 lines into "hold-file" in memory**
LOAD PHONE INTERIM	**you can DELETE it now or later**
& MERGE	**combine old and new, now 300 DATA statements**

In line 1720 and 2590, change the figure 150 to 300.
In line 1160, change the program name to PHONE LIST 300.

SAVE PHONE LIST 300	**wherever you want the finished product.**

We start with Phone List, a program on your DOS System Master disk, and double its capacity from 150 to 300 names.

This clever program actually stores names and telephone numbers in DATA statements with line numbers from 201 through 350. The two DEL statements eliminate all lines but these, which are then renumbered 351 through 500 by 1. The &HOLD command hides these lines and a LIST command at this point would show no lines. After the Phone Interim program is reloaded, the hidden lines are merged into the gap between lines 350 and 1000.

When you consider all that this involves, the process is very rapid. It's hard to see how such a significant change could have been wrought any other way, without a lot of tiresome typing. Using the techniques shown above, you can move a bunch of lines around within a program, combine two programs, and incorporate proven routines from one program to another without the error-prone step of retyping.

Some programs have lines with line numbers greater than 63999, the legal maximum. Renumber is clever enough to leave these alone, and this is probably for the best. A word of caution in this area: I once fabricated an illegal line number 65535 and spent several days looking for the mysterious cause of a number's silently changing from 2 to 2.000000007. The problem disappeared when I removed the bad line number.

As with most tools, practice improves skill. Do use the Brenumber program to its limit – it's very, very good. But, and it's a big *but*, be prudent. Save an important program before renumbering it, and don't overwrite that backup until the renumbered version is proven.

Homework Assignment: If you have an Apple II Plus with no Integer ROM Card nor Language Card, you may have no Apple Mini-Assembler. In that case, follow the instructions below. You will create a one-pass assembler which will be of use in future columns in this series. Please note that "CTRL-Y" means hold down the CTRL key and type "Y". In the lines where this is used, a space is shown for clarity only; do not type any spaces in those two lines!

How To Make A Mini-Assembler If You Have An Apple II Plus

Take a diskette to an Apple which has both Integer BASIC and Programmer's Aid.

If a 16K RAM card is installed, boot the System Master diskette.

```
]INT
>CALL -151
*D4D5G
*6000:4C 98 60
*6003<F500.F63C CTRL-Y *
*6003<F500.F63C CTRL-Y
*BSAVE MINI-ASSM,A$6000,L$140
(THANK THE NICE APPLE.)
```

Program 1.

```
RUN RENUMBER
(PRESS RETURN WHEN INVITED.)
CALL -151
8DE0: A9 8E 85 70 85 74 8D F7
8DE8: 03 A9 00 85 6F 85 73 8D
8DF0: F6 03 A9 4C 8D F5 03 20
8DF8: 6C D6 4C D0 03 4C D0 03
90DE: 20 F0 95
95ED: 30 8D 28
95F0: 20 B5 94 08 C9 20 F0 F7
95F8: 28 60
94D4: BC     <---BUG FIXER!!!
(INSERT DISK TO HOLD RESULT.)
BSAVE BRENUMBER,A$8DE0,L$820
```

Program 2.

```
RUN RENUMBER
(PRESS RETURN WHEN INVITED.)
CALL -151
8DE0: A9 8E 85 70 85 74 8D F7
8DE8: 03 A9 00 85 6F 85 73 8D
8DF0: F6 03 A9 4C 8D F5 03 20
8DF8: 6C D6 4C D0 03 4C D0 03
90DA: 20 F0 95
95ED: 30 8D 28
95F0: 20 B2 94 08 C9 20 F0 F7
95F8: 28 60
94D1: BC     <---BUG FIXER!!!
(INSERT DISK TO HOLD RESULT.)
BSAVE BRENUMBER,A$8DE0,L$820
```

©

These three short Applesoft programs show you how to change line numbers in order to delete and create undeletable lines.

Undeletable Lines, Revisited

P. Kenneth Morse
Augusta, GA

Michael P. Antonovich described (**COMPUTE!**, October 1981, #17) a method of using the Apple's monitor to enter Applesoft program statements that could not be easily deleted using the Applesoft DEL command. He indicated that a way to get rid of such lines was to change the end-of-program pointer in $69-6A (115-116, decimal).

There are, however, at least two other general approaches to deleting "undeletable" lines:

(1) Change the line number back to a deletable number. This may be done by using the monitor (or POKE statements) to modify the number of a specific line, or by simply running Apple's Renumber program. Once a deletable line number has been achieved, DEL will complete the job. Readers may find the program text file Deletable (see Program 1) helpful in quickly gaining control of undeletable lines.

(2) LIST the deletable portion of the program to a new text file, clear memory with a NEW command, and then EXEC the text file. The undeletable lines will have vanished!

Mr. Antonovich's approach (changing the end-of-program pointer) and the text file approach (#2 above) will work only when the undeletable lines are at the end of the program. However, undeletable lines may also be placed at the beginning of the program (where they inhibit LISTings beginning at specific line numbers) or in mid-program. For example:

```
10 PRINT "THIS IS ";
20 PRINT "A TEST"
```

may be converted to

```
65535 PRINT "THIS IS ";
20 PRINT "A TEST"
```

by entering

```
POKE 2051,255:POKE 2052,255
```

in immediate execution mode, and the program will RUN and LIST, but you cannot RUN, LIST or GOTO either 20 or 65535 as specific line numbers. However, it is not practical to make the opening lines undeletable, since the program would then work only for the trivial case of a program with no GOTOs or GOSUBs! To test this, enter:

```
NEW
10 PRINT "THIS IS ";
20 PRINT "A TEST "
30 GOTO 50
40 STOP
50 PRINT "IT WAS A SUCCESS"
```

and change line 10 to line 65535 as above. The program will not be able to find line 50!

Secondly, it is not necessary to key in an entire line through the monitor to achieve an undeletable line number. Programs 2 and 3 below provide Applesoft and Integer BASIC programs that will change specified line numbers to the undeletable value of 65535.

Deletable (Lines 1-8 In Program 1)

RUNning Program 1 creates a program text file, Deletable, which may be EXECed to convert undeletable Applesoft lines to a deletable range (63000-63999). Deletable will renumber up to 1000 undeletable lines per run. Once the line numbers are in the deletable range, DEL will finish the job.

Line 0 is a temporary line, used to create the text file Deletable by LISTing to the file lines 1-8. When Deletable is RUN, line 2 sets the value of the high and low bytes to be POKEd as the new deletable line number. Line 3 initializes L1, the line address, as the start-of-program address stored in bytes 103-104 (decimal). Line 4 calculates CL, the line number being tested, and determines if it is undeletable (i.e., at least 64000). If the value of CL (line 4) is undeletable, deletable values are POKEd (line 5), the POKE values are incremented, and control is passed to line 6. When all line numbers have been tested (or 1000 lines have been made deletable), Deletable deletes itself!

Deletable may also come in handy in case of a bombed Applesoft program caused by an inadvertent POKE which created an illegal line number. However, if the pointer to the next line was bombed, Deletable will not be able to help.

To use Deletable:

— Key in (and SAVE) Program 1
— RUN (this will create Deletable)
— LOAD the program containing the undeletable lines
— EXEC Deletable
— RUN

Applesoft (Lines 61800-61970 In Program 2)

The program will renumber as 65535 all lines between 62000-63999 and then delete itself. Here's how it works.

Beginning with the first line currently in memory, the program calculates CL, the current line number (line 61920). If CL is less than 62000, then the address (L1) of the next line number is calculated in line 61940, and the program recycles to 61920. If the number is at least 62000, the address of the line number is saved in the L() array, and a test is made to determine if the end of the lines to be renumbered has been reached.

If not, the program returns to 61920 to test the next line number. If it is the final line, it then renumbers each line referenced in the L() array to 65535, and DELetes Applesoft Permanent Notice. If you expect to renumber more than ten lines, you will need to DIM L(), either as a direct command or by inserting a DIM statement in the program.

To use Applesoft Permanent Notice:

— Key in and SAVE Program 2
— RUN (this creates Applesoft Permanent Notice)
— NEW or LOAD a program
— Enter lines numbered 62000-63999 which you want to be made permanent
— EXEC Applesoft Permanent Notice
— RUN 61800

Integer (Lines 31000-31170 in Program 3)

Integer Permanent Notice operates in somewhat similar fashion, but the lines to be renumbered should be in the range 32000-32767 (remember: Integer BASIC doesn't like numbers greater than 32767). Lines 31070-31090 determine L1, the address of the line number to be tested. CL, the line number itself, is calculated in 31100 and tested in 31110. If CL is 32000 or greater (line 31120), then L1 is tested (line 31130) to see if the final line has been tested. If not, the address L1 is stored in the array ADD(L). L is incremented, and control shifts to line 31120. When all line numbers have been tested, the value 255 (line 31160) is POKEd into both bytes of each address stored in ADD(). ADD() is currently DIMensioned at 10; this may be changed to renumber more than ten lines to undeletable status.

To use Integer Permanent Notice:

— Key in and SAVE Program 3
— RUN
— NEW or LOAD a program file
— Enter lines to be made permanent. Number them between 32000 and 32767
— EXEC Integer Permanent Notice
— RUN 31000

Program 1: Deletable

```
0 D$ =  CHR$ (4): PRINT D$"OPEN DELETABLE": PRINT D$"WRITE
   DELETABLE": LIST 1 - 8: PRINT D$"CLOSE DELETABLE": END
1  REM

       SAVE 'DELETABLE' BEFORE RUNNING!


2 HI = 246:LO = 24: REM  VALUE IS 63000
3 L1 =  PEEK (103) + 256 *  PEEK (104):L = 0
4 CL =  PEEK (L1 + 2) + 256 *  PEEK (L1 + 3): IF CL < 64000
   THEN 6
5  POKE L1 + 2,LO: POKE L1 + 3,HI:LO = LO + 1: IF LO > 255
   THEN LO = 0:HI = HI + 1
6 L1 =  PEEK (L1) + 256 *  PEEK (L1 + 1): IF L1 = 0 THEN 8
7  IF 256 * HI + LO < 64000 THEN 4
8  DEL 1,8
```

Program 2: Applesoft Undeletable

```
1 D$ =  CHR$ (4):F$ = "APPLESOFT PERMANENT NOTICE": PRINT
   D$"OPEN"F$: PRINT D$"WRITE"F$: LIST 61800,61970: PRINT
   D$"CLOSE": END
61800  REM  --------------------
61810  REM    'PERMANENT NOTICE'
61820  REM       BY KEN MORSE
61870  REM  -------------------
61875  TEXT : HOME : PRINT "'REM' LINES FOR PERMANENT
   NOTICES SHOULDBE NUMBERED 62000 OR HIGHER, AND SHOULD
   BE THE HIGHEST NUMBERED LINES IN THE    PROGRAM"
61880  PRINT : INPUT "MAKE PROGRAM LINES PERMANENT
   BEGINNING  AT 62000 THROUGH ";LL
61890  IF LL < 62000 THEN 61880
61900 FL = 62000
61910 L1 =  PEEK (103) + 256 *  PEEK (104):L = 0
61920 CL =  PEEK (L1 + 2) + 256 *  PEEK (L1 + 3): IF CL =
   > FL THEN 61950
61940 L1 =  PEEK (L1) + 256 *  PEEK (L1 + 1): GOTO 61920
61950 L(L) = L1 + 2:L1 =  PEEK (L1) + 256 *  PEEK (L1 +
   1):CL =  PEEK (L1 + 2) + 256 *  PEEK (L1 + 3): IF L1 >
   0 THEN L = L + 1: GOTO 61950
61960  FOR J = L TO 0 STEP  - 1: POKE L(J),255: POKE L(J) +
   1,255: NEXT J
61970  DEL 61800,61970: END
```

Program 3: Integer BASIC Undeletable

```
1 D$ = "": REM    CTRL-D
2  PRINT D$;"OPEN INTEGER PERMANENT NOTICE": PRINT
   D$;"WRITE INTEGER PERMANENT NOTICE": LIST 31000,31170:
   PRINT D$;"CLOSE"
31000  REM     --------------------
31010  REM       'PERMANENT NOTICE'
31020  REM       FOR INTEGER BASIC
31030  REM         BY KEN MORSE
31040  REM     -------------------
31050  DIM ADD(10)
31060 FL = 31999:L = 0
31070 L1 =  PEEK (202):L2 =  PEEK (203)
31080  IF L2 > 127 THEN L2 = L2 - 256
31090 L1 = L1 + 256 * L2
31100 CL = ( PEEK (L1 + 1)) + 256 * ( PEEK (L1 + 2))
31110  IF CL > FL THEN 31130
31120 L1 = L1 +  PEEK (L1): IF CL <  = FL THEN 31100
31130  IF L1 >  PEEK (76) + 256 * ( PEEK (77) - 256) THEN
   31160
31140 ADD(L) = L1
31150 L = L + 1: GOTO 31120
31160  FOR J = L TO 0 STEP  - 1: POKE ADD(J) + 1,255: POKE
   ADD(J) + 2,255: NEXT J
31170  END
```

©

Restore (N) In Applesoft

Michael Erperstorfer
Vienna, Austria

The usual RESTORE statement in Applesoft simply resets the data list pointer to the first occurrence of a DATA statement in an Applesoft program, though in some applications it would be necessary to READ from a distinct DATA line. With a small machine language program, it is rather easy to build a RESTORE with a parameter.

This is done with the ampersand (&) command. This symbol, when executed as an instruction, causes an unconditional jump to memory location $03F5. At location $03F5 there must be a JMP instruction to your machine language program, which is then terminated with an RTS instruction to pass control back to Applesoft.

The syntax of RESTORE(N) with the ampersand is &N where N is an integer in the range 0-65535. If there is no line number N, the data list pointer will be set to the next DATA line in the program. If there are no more DATA lines, an OUT OF DATA error message will be displayed. Before the first use, the machine language must be linked with CALL 768.

```
0000-                             ;RESTORE(N)...&N
0000-                             ;
0300-   A9 0B       LDA   #$0B    ;LOAD LB OF &-JMP.
0302-   8D F6 03    STA   $03F6
0305-   A9 03       LDA   #$03    ;LOAD HB.
0307-   8D F7 03    STA   $03F7
030A-   60          RTS           ;END OF INITIALIZATION.
030B-   20 67 DD    JSR   $DD67   ;EVALUATES NUMBER AND
                                  ;STORES IT IN FP-AAC #1.
030E-   20 52 E7    JSR   $E752   ;CONVERTS FP-ACC #1 INTO
                                  ;2BYTE INT ($50,$51)
0311-   20 1A D6    JSR   $D61A   ;SEARCHES FOR LINE#
                                  ;(IN $50,$51).
0314-   38          SEC           ;$9B,$9C POINT TO LINK
                                  ;FIELD OF DESIRED LINE.
0315-   A5 9B       LDA   #$9B    ;LOAD DATA LIST POINTER
                                  ;($7D,$7E)
0317-   E9 01       SBC   #$01    ;CONTENTS OF $9B,$9C-1.
0319-   85 7D       STA   $7D
031B-   A5 9C       LDA   $9C
031D-   E9 00       SBC   #$00
031F-   85 7E       STA   $7E
0321-   60          RTS           ;RETURN TO BASIC.
```

BASIC loader for machine language program:

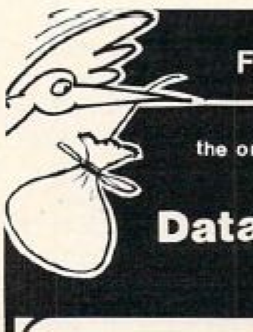
```
10 FOR I = 768 TO 801: READ V: POKE I,V: NEXT

15 CALL 768
20 DATA 169,11,141,246,3,169,3,141,247,3,96,3
    2,103,221,32,82,231,32,26
30 DATA 214,56,165,155,233,1,133,125,165,156,
    233,0,133,126,96
```

The ML routine can now be saved either on disk with BSAVE RESTORE(N),A$300,L$22 or on tape with 300.321W.

A simple example for the use of &N:

```
10 PRINT CHR$(4)"BRUN RESTORE(N)"
20 INPUT "LINENUMBER: ";LN
30 & LN
40 READ L
50 PRINT "HERE IS LINE #"L
60 GOTO 20
100 DATA 100
110 DATA 110
120 DATA 120
130 DATA 130
140 DATA 140
150 DATA 150
```

©



FIRST BORN IN 1978!

the original & continuously updated

CCA Data Management System

Now Available For Atari Computers	$ 99.50
For Apple Computers	150.00
For CPM Based Computers	225.00

CCA Data Management System

Uses

- Business
  Accounts Receivable
  Accounts Payable
  Inventories
  Billing
  Lists and Rosters
- Home Phone Lists
  Budgets, Hobbies

Features And Capabilities

- Long record lengths
- Up to 24 fields per record
- **Not Copy Guarded**
- Alpha numeric items
- Numeric only items
- Add, update, scan, etc. files
- 10-Level sort ascending, descending, allows alphabetizing data file.

• Contact your local dealer for details or write us for our catalog

DIVISION OF CUSTOM ELECTRONICS, INC.
SOFTWARE
238 Exchange St., Chicopee, Massachusetts 01013
(413) 592-4761
Mastercard & VISA Accepted
• Dealer And Distributor Inquiries Invited
• Closed Mondays — Open Daily 'Til 5:30 — Fridays 'Til 8

COMPUTER PL

ATARI NECESSITIES

ATARI 400 16k		244.95
ATARI 800 **48k!!**		659.95
PROGRAMMER KIT		59.95
VALFORTH	(d)	39.95
GENERAL UTILITY PACKAGE	(d)	35.95
PLAYER MISSILE, CHAR. & SOUND	(d)	35.95
DISPLAY FORMATTER	(d)	29.95
TURTLE GRAPHICS & FP ROUTINES	(d)	39.95
TEXT COMPRESSION & FORMATTING	(d)	29.95
HEAVY DUTY BINDER		11.95
ALL SEVEN ITEMS ABOVE	(d)	179.95
TRICKY TUTORIALS 1, 2, 3, 4, 6, 8 Ea	(c,d)	16.95
TRICKY TUTORIALS 5, 7 Ea	(c,d)	24.95
TRICKY TUTORIALS 9, 10, 11, 12		CALL
DE RE ATARI	(b)	17.95
TECHNICAL REFERENCE NOTES	(b)	24.95
DOS UTILITIES SOURCE LISTING	(b)	5.95
INSIDE ATARI DOS	(b)	16.95
YOUR ATARI COMPUTER	(b)	14.95
ATARI GAMES & RECREATIONS	(b)	12.95
ATARI SOUND & GRAPHICS	(b)	8.95
PICTURE THIS (ATARI PILOT)	(b)	12.95

COMPUTER PLACE

P.O. Box 6007-169 1698 Market St.
Redding, CA 96001 Ph. 916-221-1312

COMPUTER PLACE is a storefront operation. We have the largest selection of items for Atari, Apple, Commodore and Radio Shack computers that you are ever likely to see. Try us for obscure and hard to get items that your local dealer might not stock. Mail Order Terms: Cashier's check or M.O. Charge cards add 4%. Add 3% shipping and handling (minimum $2.50)-UPS ground. No. C.O.D.'s. CA residents add 6%sales tax. Prices subject to change without notice.

TCE Presents

Holiday SOFTWARE Specials

PROGRAM	COMPUTER	TCE	LIST
WORDSTAR	1	$199	$375
SUPER TEXT II	1	99	150
MICROTEK 16K RAM CARD	1	89	120
TEXT WIZARD	2	69	100
HOME ACCOUNTANT	1	52	75
GRAPHICS MAGICIAN	1	39.99	59.95
DEADLINE	1,2	33.99	49.95
WIZARDRY	1	33.99	49.95
MASTERTYPE	1,2	26.99	39.95
ULTIMA	1,2	26.99	39.95
ZORK I	1,2	26.99	39.95
BANDITS	1,2	22.99	34.95
CHOPLIFTER	1,2	22.99	34.95
FROGGER	1,2	22.99	34.95
KNIGHT OF DIAMONDS	1	22.99	34.95
POOL 1.5	1,2	22.99	34.95
SERPENTINE	1,2	22.99	34.95
STAR BLAZER	1,2	21.99	31.95
APPLE PANIC	1,2	$19.99	29.95
CLOWNS & BALLOONS	2		29.95
CROSSFIRE	1,2		29.95
PACIFIC COAST HWY	2		29.95
MASTER BLASTER	1,2		29.95
SPACE EGGS	1,2		29.95
SNAKE BYTE	1,2		29.95
SNEAKERS	1,2		29.95
TRACK ATTACK	1,2		29.95
UTILITY CITY	1		29.50

APPLE=1 ATARI=2 REGISTERED TRADEMARKS Holiday Specials through 12/31/82

ELEPHANT DISKS
SS-SD 19.99/10
SS-DD 23.99/10
DS-DD 29.99/10

SIGNALMAN MODEM
$85.99 LIST 99.95

BMC MONITORS
$89 12" GREEN
$269 13" COLOR

FLIP N FILE $22.95

HEAD CLEANER $22.95

C.ITOH PROWRITER
$409 Parallel List $795
$629 Serial List $895

HOLIDAY SPECIALS
Send for our full list of software & accessories on special through Christmas.

FREE Catalog of APPLE-ATARI-IBM-TRS-80-VIC Software

The Computer Express

M-F 10 AM - 9 PM
Sat 10 AM - 6 PM
313 528-1554

P.O. Box 569, Dept B
Troy, MI 48099
FREE CATALOG

VISA

Add $2.00 for shipping. MI residents add 4% tax. Money orders, checks (allow 10 days) accepted. MC/VISA accepted with card # & expir. date. Add $1.50 for C.O.D. Prices subject to change.

A Graphics Plot For The Epson MX-80 Printer

William L. Osburn
Wyoming, DE

For those Atari owners with an Epson MX-80 printer and the Macrotronics Parallel Printer Interface, here is a short BASIC routine which will copy a graphics mode 7 display onto the printer.

Before running this routine, X$ must be DIMensioned to 80 characters. The graphics you want copied must already be displayed on the screen.

```
5200 REM SET UP GRAPH PRINTER PLOT
5210 LPRINT CHR$(27);CHR$(65);CHR$(131);
        CHR$(27);CHR$(50);CHR$(15)
5230 FOR I=0 TO 159
5240 FOR J=0 TO 79:POSITION I,J:GET #6,A
5250 IF A=0 THEN X$(79-J+1,79-J+1)=" ":
        GOTO 5270
5255 IF A=1 THEN X$(79-J+1,79-J+1)=".":
        GOTO 5270
5260 X$(79-J+1,79-J+1)="*"
5270 NEXT J
5280 LPRINT X$
5290 NEXT I
5295 LPRINT CHR$(7);CHR$(7)
5300 END
```

Line 5210 sets up the horizontal and vertical spacing of the Epson MX-80. The commands CHR$(27); CHR$(65); CHR$(131); CHR$(27); CHR$(50) set the vertical line spacing to 3/72 of an inch. The line spacing can be set to any 1/72 of an inch. CHR$(15) sets the horizontal printing to 132 characters per line. Lines 5230 and 5240 direct the pixel scan of the screen. The command LOCATE I,J,A can be used in place of POSITION I,J: GET #6, A in line 5240. The returned value of variable A will be either 0,1,2, or 3, depending on the COLOR exp used for that pixel. Lines 5250, 5255 and 5260 set the X$ array for printing depending on the value returned for A. In this case I used "." for the border and "*" for the curve. The border and the curve were in different colors. Line 5280 prints the string array X$ (x=I, y=0 to 79). Line 5295 rings the MX-80 buzzer after the printing is done. The plot takes about nine minutes to complete and this allows me to do something else.

The result is a copy of the screen onto paper. The routine rotates the plot 90° clockwise in order to print. ©

commodore

VIC 20 . . . . . $179

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

VIC Software

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

United Microware

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

Creative Software

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

American Peripherals

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

FRIENDLY SERVICE

APPLE SOFTWARE NEW

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

*** SPECIALS OF THE MONTH ***

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

Computer Outlet

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

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

APPLE SPECIALS

Entertainment

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

Business

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

Peripherals

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

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

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

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

Inside Apple

Apple Computer Inc., 20525 Mariani Avenue, Cupertino, California 95014 Vol. 1 No. 2

For the authorized Apple dealer nearest you, call 800-538-9696 (800-662-9238 in California.)

Fruitful Connections.

There are more people in more places making more accessories and peripherals for Apples than for any other personal computer in the world.

Thanks to those people—in hundreds of independent companies—you can make the humblest 1978 Apple II turn tricks that are still on IBM's Wish List for 1984.

But now we're coming out with our very own line of peripherals and accessories for Apple® Personal Computers.

For two very good reasons.

First, compatibility. We've created a totally kluge-free family of products designed to take full advantage of all the advantages built into every Apple.

Second, service and support.

Now the same kindly dealer who keeps your Apple PC in the pink can do the same competent job for your Apple hard-disk and your Apple daisywheel printer.

So if you're looking to expand the capabilities of your Apple II or III, remember:

Now you can add Apples to Apples.

A joy to behold.

The new Apple Joystick II is the ultimate hand control device for the Apple II.

Why is it such a joy to use?

With two firing buttons, it's the first ambidextrous joystick—just as comfortable for lefties as righties.

Of course, it gives you 360° cursor control (not just 8-way like some game-oriented devices) and full X/Y coordinate control.

And the Joystick II contains high-quality components and switches tested to over 1,000,000 life cycles.

Which makes it a thing of beauty. And a joystick forever.

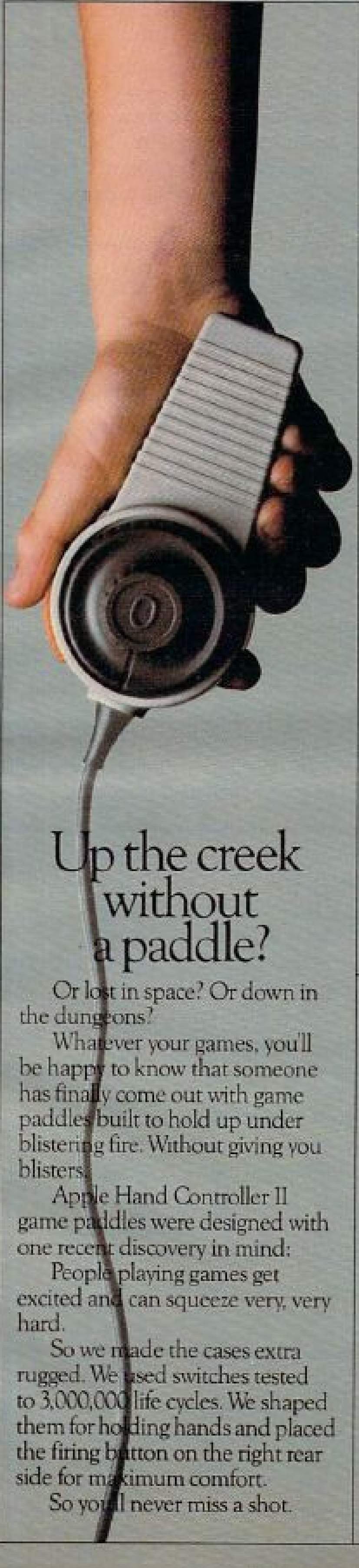
Gutenberg would be proud.

Old Faithful Silentype® has now been joined by New Faithfuls, the Apple Dot Matrix Printer and the Apple Letter Quality Printer.

So now, whatever your budget and your needs, you can hook your Apple to a printer that's specifically designed to take advantage of all the features built into your Apple. With no compromises.

The 7x9 Apple Dot Matrix Printer is redefining "correspondence quality" with exceptional legibility. With 144x160 dots per square inch, it can also create high resolution graphics.

The Apple Letter Quality Printer, which gets the words out about 33% faster than other daisywheel printers in its price range, also offers graphics capabilities. See your authorized Apple dealer for more information and demonstrations. Because, unfortunately, all the news fit to print simply doesn't fit.



Up the creek without a paddle?

Or lost in space? Or down in the dungeons?

Whatever your games, you'll be happy to know that someone has finally come out with game paddles built to hold up under blistering fire. Without giving you blisters.

Apple Hand Controller II game paddles were designed with one recent discovery in mind:

People playing games get excited and can squeeze very, very hard.

So we made the cases extra rugged. We used switches tested to 3,000,000 life cycles. We shaped them for holding hands and placed the firing button on the right rear side for maximum comfort.

So you'll never miss a shot.

A storehouse of knowledge.

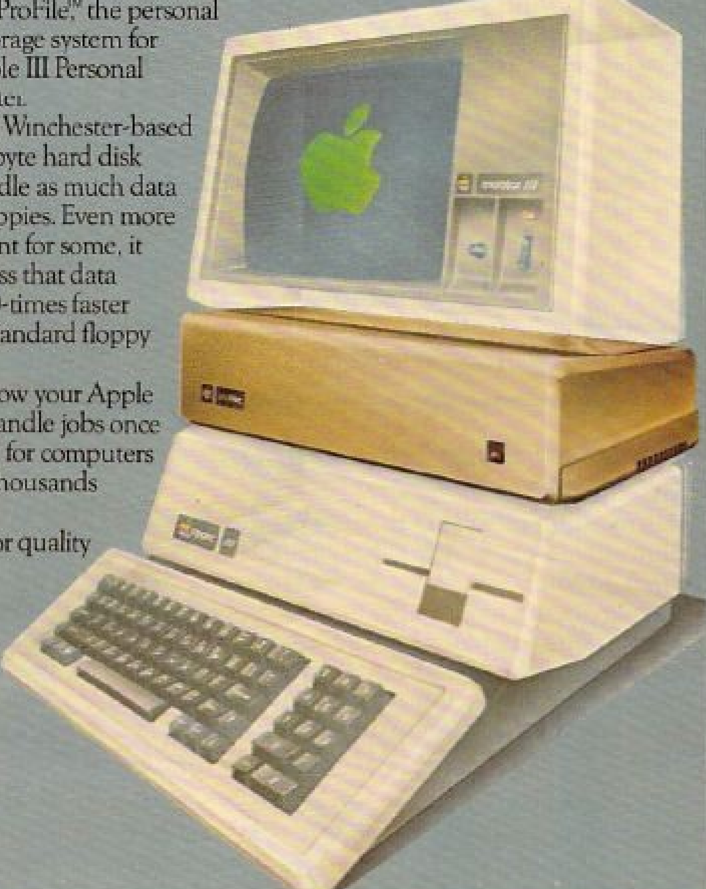
If you work with so much data or so many programs that you find yourself shuffling diskettes constantly, you should take a look at Apple's ProFile,™ the personal mass storage system for the Apple III Personal Computer.

This Winchester-based 5-megabyte hard disk can handle as much data as 35 floppies. Even more important for some, it can access that data about 10-times faster than a standard floppy drive.

So now your Apple III can handle jobs once reserved for computers costing thousands more.

As for quality and reliability, you need only store one word of wisdom:

Apple.



Launching pad for numeric data.

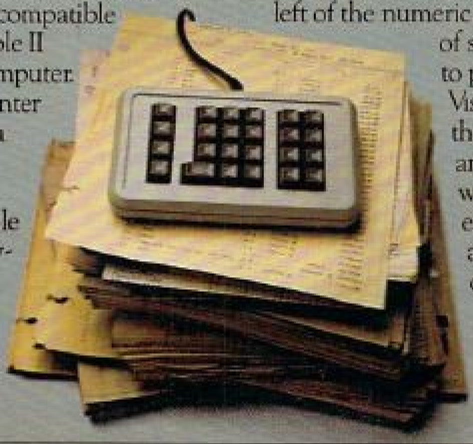
Good tidings for crunchers of numerous numbers:

Apple now offers a numeric keypad that's electronically and aesthetically compatible with the Apple II Personal Computer. So you can enter numeric data faster than ever before.

The Apple Numeric Keypad II has a standard calculator-style layout. Appropriate, because unlike some other keypads, it can actually function as a calculator.

The four function keys to the left of the numeric pad should be of special interest to people who use VisiCalc.® Because they let you zip around your work sheet more easily than ever, adding and deleting entries.

With one hand tied behind your back.



VisiCalc is a registered trademark of VisiCorp, Inc.

If you're ready to learn intermediate BASIC programming

STEP BY STEP TWO IS READY FOR YOU!

In Each Parentheses, Choose One:

If you're one of the (happy, smart, ecstatic) thousands who learned (quickly, clearly, non-boringly) to use their APPLE computers with our beginners' BASIC tutorial, *The New Step By Step*, then we (know, believe, swear) that *Step By Step Two* is ready to (ease, breeze, squeeze) you into the (advanced, intermediate, grown-up) world of PEEK and POKE, hexidecimal numbers, concatenations, and (much, much, much) more.

On the other (hand, foot) if you didn't (grow with, thrill to, involve the whole family in) the PDI (learning, hands-on, fun) experience, here's what you missed:

- Teaching techniques that teach
- Involvement you enjoy
- Sounds that spur you on
- Graphics that simplify the complex
- Animation that makes this tutorial a stimulating experience.
- A friendly voice that guides you to course completion

The *Step By Step Two* program works this way:

- the computer program sets up screen displays or sample programs for you.
- the cassette voice tells you what's happening.
- you (deal with, figure out, guess at) the answer.
- the computer (praises, pans) your work.
- you (peruse, plunge into, practice in) the Work Book.
- after each lesson, you're (quizzed, queried, questioned).
- you're then (prepared, practiced, primed) for the next lesson.
- the final exam reveals all (superstars, slackers).

There's lots to learn in *Step By Step Two:*

- PEEK & POKE
- Default values
- Memory map
- CALL program
- CHR$ and ASC functions
- control characters
- ASCII codes
- RAM vs ROM
- hexidecimal numbers
- machine monitor
- string logic
- string arrays
- high resolution graphics
- screen memory

But don't take our (word, words, wordiness) about how (good, great, grand) the *Step By Step* method is. Listen to our (critics, reviewers, friends):

"If you want to learn BASIC or would like a little guidance and encouragement added to what you already know, then the way to go is *Step By Step.*" —Softalk

"The *Step By Step* approach is the next best thing to having an experienced programmer by your side... *Step By Step* is a superb example of a successful blend of various media. The teaching principles are sound, the execution is virtually flawless, and the whole thing works." —Popular Computing.

If you want to move ahead in BASIC programming, the next (simple, logical, shrewd) step is *Step By Step Two.*

Step By Step Two is available at fine retail stores or direct from PDI for $89.95 plus $3.00 shipping and handling. (The package includes back-up discs.)

P.S. If you've yet to take your first step into BASIC, it's time to get *The New Step By Step*! Same great tutorial techniques, for $79.95 plus $3.00 shipping and handling.

PDI
Program Design, Inc.
11 Idar Court
Greenwich, CT. 06830
203-661-8799

*Apple is a trademark of Apple Computer Corp.

Easy Apple Disk Space Messages

Beirne L. Konarski, Kent, OH

As diskettes fill up, it is useful to know how much room is left on them. Two methods exist for finding this number. One is to get a calculator or pencil and paper and add the figures. This seems pointless, since the computer is supposed to do those tasks. The alternative is to run the FID program from the system master. This is a nuisance, though, because you often change diskettes.

Since many people incorporate a CATALOG command into their greeting program, this would be the most useful place for a sector-counting subroutine. The *DOS Manual* contains two sections which help to solve this problem. The first is the RWTS (Read or Write a Track and Sector) machine language program (p. 94). The second is the listing of sector allocation (p. 129). The RWTS subroutine can be used to read the sectors containing the catalog and tally the amount of sectors used.

The diskette directory is located in track $11. It contains 15 sectors of catalog information, each holding the names and sizes of seven programs. The program reads one sector at a time beginning with sector $F and places it into the memory range beginning at $2000. The sector is then checked before the next one is read from the disk.

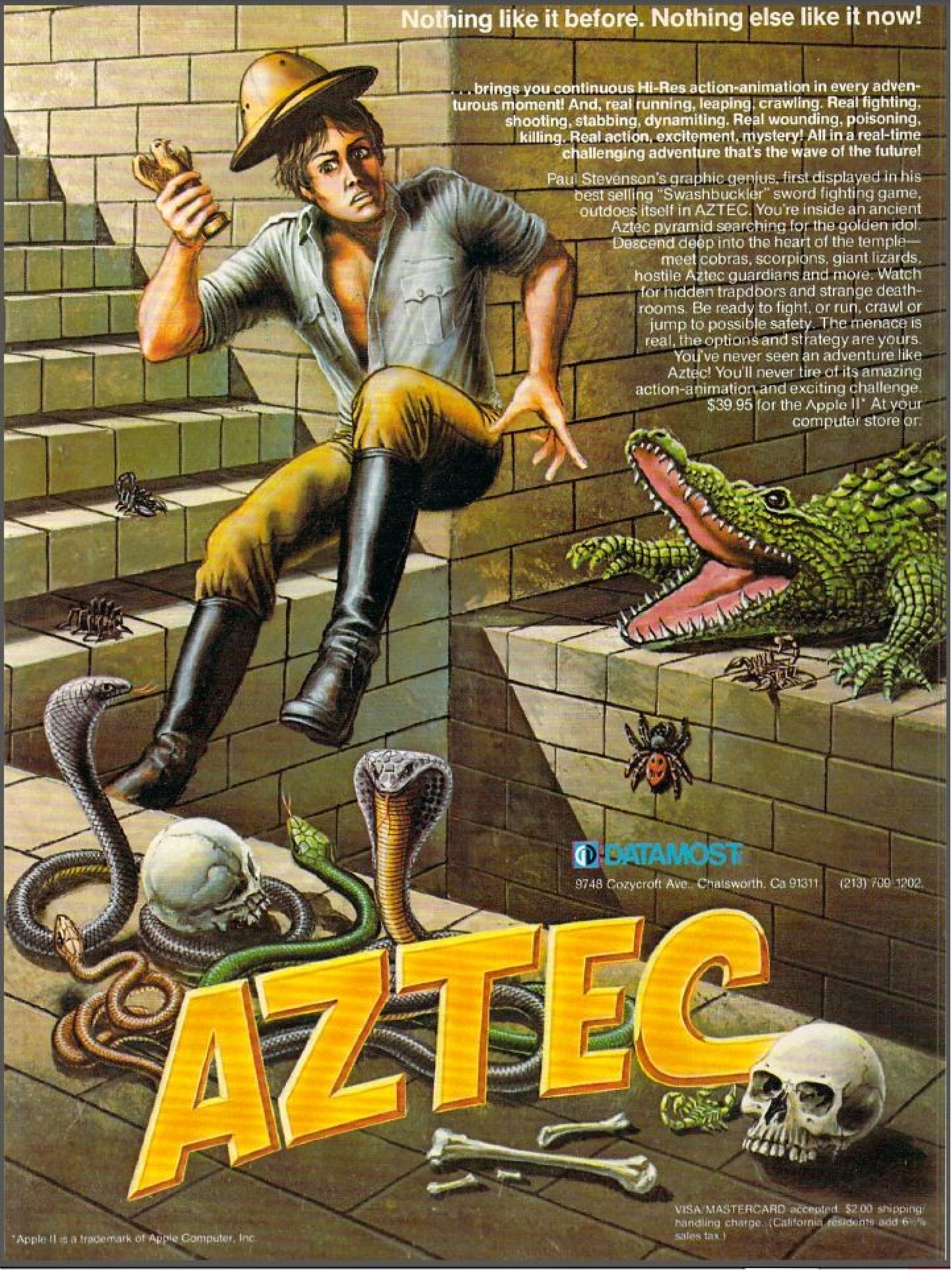
The seven program titles are checked for two things. The first is to see if the program is current. When a program is deleted, its catalog entry is not erased, but instead the first byte of the listing becomes $FF. If the program is current, then the bit containing the length is added to the running total. The Least Significant Byte of the total is stored in location $6074, or 24692. If there is a carry, it is plaçed in $6075. When all seven listings have been checked, the next sector is loaded, and the process continues until all fifteen sectors are read. The computer then returns to BASIC.

The rest of the BASIC program PEEKs the two locations containing the sum of the sectors used. The Most Significant Byte is multiplied by 256 and added to the LSB. This total is displayed and then subtracted from 496 to give the figure for the space remaining.

The program reads the DATA lines and enters the numbers beginning at $6000. It displays the CATALOG, then gives the results of the count. It can be substituted for your HELLO program, with your own personal touches like name and date placed before or after line 20, or after line 110.

```
10 D$ =  CHR$ (4)
20  PRINT D$;"CATALOG"
30  FOR K = 24576 TO 24694
40  READ A
50  POKE K,A
60  NEXT
70  CALL 24576
80 X =  PEEK (24692):Y =  PEEK (24693)
90 Z = X + 256 * Y
100  PRINT : PRINT "SECTORS USED: ";Z
110  PRINT "SECTORS REMAINING: ";496 - Z
120  END
130  DATA   169, 96, 160, 76, 32, 217, 3, 17
     3, 11, 32
140  DATA   201, 255, 240, 3, 32, 99, 96, 23
     8, 118, 96
150  DATA   173, 118, 96, 201, 7, 208, 28, 1
     69, 0, 141
160  DATA   118, 96, 169, 44, 141, 104, 96,
     169, 11, 141
170  DATA   8, 96, 173, 2, 32, 201, 0, 240,
     25, 141
180  DATA   81, 96, 76, 0, 96, 173, 104, 96,
      105, 35
190  DATA   141, 104, 96, 173, 8, 96, 105, 3
     5, 141, 8
200  DATA   96, 76, 7, 96, 96, 0, 1, 96, 1,
     0
210  DATA   17, 15, 94, 96, 0, 32, 0, 0, 1,
     0
220  DATA   0, 96, 1, 0, 0, 1, 239, 216, 0,
     24
230  DATA   173, 116, 96, 109, 44, 32, 141,
     116, 96, 144
240  DATA    3, 238, 117, 96, 96, 0, 0, 0, 0
```

©



Nothing like it before. Nothing else like it now!

. . . brings you continuous Hi-Res action-animation in every adventurous moment! And, real running, leaping, crawling. Real fighting, shooting, stabbing, dynamiting. Real wounding, poisoning, killing. Real action, excitement, mystery! All in a real-time challenging adventure that's the wave of the future!

Paul Stevenson's graphic genius, first displayed in his best selling "Swashbuckler" sword fighting game, outdoes itself in AZTEC. You're inside an ancient Aztec pyramid searching for the golden idol. Descend deep into the heart of the temple—meet cobras, scorpions, giant lizards, hostile Aztec guardians and more. Watch for hidden trapdoors and strange death-rooms. Be ready to fight, or run, crawl or jump to possible safety. The menace is real, the options and strategy are yours. You've never seen an adventure like Aztec! You'll never tire of its amazing action-animation and exciting challenge. $39.95 for the Apple II* At your computer store or:

DATAMOST

9748 Cozycroft Ave., Chatsworth, Ca 91311 (213) 709-1202.

AZTEC

VISA/MASTERCARD accepted. $2.00 shipping/handling charge. (California residents add 6½% sales tax.)

*Apple II is a trademark of Apple Computer, Inc.

A New, Improved Computer Friend For Your Apple

Fred D'Ignazio, Associate Editor

This column catches me in the midst of a move from Chapel Hill, North Carolina, to Roanoke, Virginia. Most of my dozen computers are still in Chapel Hill being looked after by a trusted babysitter. My roof leaks, my shower floods the bathroom floor, my study is buried in boxes, and Catie and Eric just came down with ear infections.

In addition to their sore ears, Catie and Eric are going through something you might call "computer-starvation shock." They think they are still a multi-computer family, and they brag about it to all their friends. They gather a horde of neighborhood kids with the promise of a dozen beeping, flashing computers. They climb the stairs, peek into my study, and what do they see? One lone computer. A rather sad-looking machine, vintage 1977. It doesn't talk, doesn't make pictures, doesn't play music. What a letdown!

Thanks, Chuck!

It's times like these when you readers come in handy. Thanks to one reader – Chuck Johnston of Manhattan Beach, California – I can still provide you with a useful column this month.

Chuck recently sent me a program he wrote that modifies my "Talking Head" program for the Apple. In my opinion, Chuck's program is a substantial improvement on the original version. It's exactly the kind of feedback I'd like to get from my readers. Thanks, Chuck!

Below is Chuck's letter and his program:

I am writing in regard to your column which appears in the September issue of **COMPUTE!** *Magazine. I found your article interesting, but the changes you suggest for the Apple II were, in my opinion, inadequate.*

The Apple is incapable of printing a reverse slash (as is this ancient typewriter), so the head shape you designed does not work. Also, you suggest deleting the sound subroutine, but it makes the program much more interesting. I have revised your program to run on the Apple and thought you might like to see it. I also failed to understand why you didn't draw the head using graphics; as you can see, the resulting animation is much more effective.

Included also is a sound driver program for the Apple in line 20, since, as we know, the Apple is only capable of rudimentary buzzes and clicks in Applesoft. It is POKEd into memory at $0300 and the POKEs in the sound subroutines are as follows: POKE 768,x (where x is a number between 1 and 255) sets the tone frequency. POKE 769,y (y also between 1 and 255) sets the tone duration. In the program enclosed I used the same values as the original program; whether it sounds the same is unlikely, but with some adjustment it could come close. Well, I hope you like the program and thank you for your time.

Chuck Johnston

```
10   REM *** POKE SOUND DRIVER INTO MEMORY
20   FOR I = 770 TO 795: READ M: POKE I,M: NEXT
```

Fred D'Ignazio is a computer enthusiast and author of several books on computers for young people. He is presently working on two major projects: he is writing a series of books on how to create graphics-and-sound adventure games. He is also working on a computer mystery-and-adventure series for young people.

As the father of two young children, Fred has become concerned with introducing the computer to children as a wonderful tool rather than as a forbidding electronic device. His column appears monthly in **COMPUTE!**.

```
40  GR : HOME
50  FOR P = 1 TO 800: NEXT
100  REM ***DIMENSION VARIABLES
120 N = 1: REM *MESSAGE POINTER
500  REM  *** FRIEND MASTER
510  GOSUB 1010: REM  *FRIEND WAKE UP
520  GOSUB 2010: REM  *FRIEND TALK
530  GOSUB 3210: REM  *STORE CHILD'S NAME
540  GOSUB 2010: REM  *FRIEND TALK
550  PRINT : PRINT : PRINT : PRINT : PRINT :
      END
1000  REM *** FRIEND WAKE UP
1010  GOSUB 5010: REM  *DRAW FACE
1020  GOSUB 5410: REM  *DRAW SLEEPING EYES
1035  FOR P = 1 TO 800: NEXT
1040  GOSUB 4000: REM  *WAKE UP BELL
1050  GOSUB 5460: REM  *DRAW OPEN EYES
1060  FOR P = 1 TO 600: NEXT
1070  GOSUB 5320: REM  *WINK EYE
1080  FOR P = 1 TO 100: NEXT
1085 M = 0: GOSUB 4820: REM  *WINK NOISE
1090  GOSUB 5460: REM  *DRAW OPEN EYES
1100  FOR P = 1 TO 800: NEXT
1110  RETURN
2000  REM *** FRIEND TALK
2005  REM  * SELECT MESSAGE
2006 N = N + 1: REM  * SET POINTER TO NEXT M
      ESSAGE
2010  READ SNUM: REM  * SNUM = NO. OF MESSAG
      ES IN SET
2015  FOR K = 1 TO SNUM
2020  GOSUB 3010: REM  *FRIEND TALK--1 SCREEN
2033  FOR P = 1 TO 1000: NEXT
2035  GOSUB 5510: REM * CLEAR MESSAGE WINDOW
2040  NEXT
2050  RETURN
3000  REM *** FRIEND TALKING--1 SCREEN
3010 PY = 21: REM  *SET VERTICAL TAB FOR TEXT
3040  READ M$
3050  IF M$ = "-1" THEN  RETURN
3051  IF M$ = "*" THEN M$ = N$
3060  VTAB PY
3070  PRINT M$;" ";: GOSUB 5250
3075  GOSUB 4810: REM  *FRIEND SOUND
3080  FOR P = 1 TO 50: NEXT : REM  *KEEP MOU
      TH OPEN
3090  GOSUB 5200: REM  *CLOSE MOUTH
3095  FOR P = 1 TO 100: NEXT : REM  *KEEP MO
      UTH CLOSED
3110  GOTO 3040
3200  REM *** FRIEND ASKS CHILD'S NAME
3210  REM
3220  VTAB 21: HTAB 10: INPUT N$
3265  FOR P = 1 TO 75: NEXT
3267  GOSUB 5510: REM * CLEAR MESSAGE WINDOW
3270  RETURN
4000  REM ***WAKE UP BELL
4010  POKE 768,30: POKE 769,105: CALL 770
4020  FOR P = 1 TO 100: NEXT
4030  POKE 768,20: POKE 769,132: CALL 770
4040  RETURN
4080  RETURN
4625  FOR P = 1 TO 15: NEXT
4800  REM *** FRIEND'S VOICE
4810 M =  INT ( RND (1) * 51) + 15
4820  FOR A = M + 25 TO M STEP  - 8
4830  POKE 768,3: POKE 769,A: CALL 770
4840  NEXT
4880  RETURN
5000  REM *** FRIEND'S FACE
5010  GR
5020  COLOR= 9: PLOT 20,10: PLOT 20,12: HLIN
      19,21 AT 11
5025  COLOR= 7: PLOT 20,13: HLIN 19,21 AT 14
      : HLIN 17,23 AT 16: HLIN 17,23 AT 18
5030  COLOR= 2: HLIN 18,22 AT 15: HLIN 17,23
      AT 17: HLIN 17,23 AT 19
5035  COLOR= 11: FOR I = 20 TO 31: HLIN 17,2
      3 AT I: NEXT : PLOT 16,24: PLOT 24,24: HL
      IN 18,22 AT 32: HLIN 18,22 AT 33
5040  COLOR= 12: HLIN 16,24 AT 34: HLIN 15,2
      5 AT 35: HLIN 15,25 AT 36: HLIN 14,26 AT
      37
5050  COLOR= 1: HLIN 19,21 AT 28
5200  REM  ***CLOSE MOUTH
5220  POKE 1852,177
5230  RETURN
5250  REM *** OPEN MOUTH
5260  POKE 1852,16
5280  RETURN
5300  REM  ***LEFT EYE WINK
5320  POKE 1467,176: POKE 1469,190
5330  FOR I = 1 TO 150: NEXT
5340  RETURN
5400  REM  ***EYES ASLEEP
5410  POKE 1467,190: POKE 1469,190
5420  RETURN
5450  REM  ***EYES AWAKE
5460  POKE 1467,176: POKE 1469,176
5470  RETURN
5500  REM  ***CLEAR MESSAGE WINDOW
5510  HOME
5550  RETURN
5600  REM *** SOUND SUBROUTINE
5610  DATA 172,01,03,174,01,03,169,04,32,168
     ,252,173,48,192,232,208,253,136,208,239
     ,206,0,03,208,231,96
6000  REM  ***MESSAGES
6010  DATA 3
6011  DATA    HI,I'M,GEB,-1
6012  DATA  YOU,TURNED,ME,ON,-1
6013  DATA       WHO'S,OUT,THERE?,-1
6020  DATA 2
6021  DATA    I'M,SO,HAPPY,-1
6022  DATA TO,SEE,YOU,*,-1
```

©

FOR THE VIC-20®

***RATED THE TOP EDUCATIONAL PROGRAM FOR THE VIC-20 BY CREATIVE COMPUTING MAGAZINE.**

- **TYPING TUTOR* — $12.95**
  If you've ever wanted to learn touch typing, this is for you! Makes learning the keyboard much easier. 4 programs on one tape teach the keys in the correct progression starting with the easy "home keys." Automatically advances to new keys as your skills develop. Ideal for beginning children, old "hunt & peck" typists, and to refresh old typing skills. Highly praised by customers: "Typing Tutor is great," "Fantastic," "Excellent," "High Quality."
- **WORD INVADERS — $10.95**
  Put excitement into your touch typing practice! Blast the invading words out of the sky before your base is destroyed. Four levels of difficulty match the letters as learned on our TYPING TUTOR program. Typing can be fun!

  SAVE, ORDER BOTH OF THE ABOVE PROGRAMS AS "TYPING TUTOR PLUS" FOR ONLY $21.95
- **FLASHCARD MAKER & FLASHCARD QUIZ — $10.95**
  2 programs. Prepare your own study material and make it easier to learn. Use for English/Foreign words, etc. Quiz program has options for study, full test, and easy learning mode. Keeps score and allows re-test of missed questions or of entire set. Includes sample data on tape with 50 States and their Capitals.

(ALL PROGRAMS ON CASSETTE TAPE AND RUN IN THE UNEXPANDED VIC)

FOR THE COMMODORE 64®

- **SPRITE DESIGNER by Dr. Lee T. Hill — $16.95**
  Save hours of work when designing sprites. Helps you create multiple sprites, copy and alter them to create views from different perspectives automatically for 3-D or animated effects. Options include: copy any of the previous sprites, reflection, rotation, translation, shearing, reverse image, merge & intersect. Saves all sprite data for easy merge into your program.

Shipping & handling $1.00 per order.
California residents add 6% sales tax.
Visa and Mastercard orders must include card number and expiration date.

ACADEMY SOFTWARE
P.O. BOX 9403
SAN RAFAEL, CA 94912 **(415) 479-4703**

Programmers. Write to our New Program Manager concerning any exceptional VIC-20 or C64 game or other program you have developed.

High Resolution Turtle Graphics

Connecting The Strobe Pen Plotter To Apple Turtle PILOT

David D. Thornburg, Associate Editor

There comes a time when most users of turtle graphics wish they could get higher resolution pictures than those shown on the display screen. The easiest way to accomplish this is to connect the computer to a graphic pen plotter. Pen plotters have been available for many years, but it is only recently that their cost has dropped to the point that they are affordable to home computer users.

Of the various low-price plotters, the Strobe Model 100 has a price of under $800 (including Apple interface card and software), and has a resolution of 0.002 inches in both axes. It uses inexpensive fine-point pens from the corner drugstore, and plots on plain 8½ x 11 paper. With special pens, it can also plot directly onto plastic sheets for overhead transparencies.

While Strobe provides several application packages for various business and other graphic applications, the plotter can also be interfaced to any program written in Applesoft BASIC. In order to use the plotter with your own programs, you must first load the printer driver program (supplied). Since this program resides just above memory location 35071 and is executed with the Applesoft CALL command, I have not found a way to use this plotter directly from Logo. Anyone who has solved this problem is invited to write about it!

Modify PILOT

Devout turtlers need not feel depressed, however, since the Turtle PILOT language by Alan Poole (published in the September 1982 issue of **COMPUTE!**) is written in Applesoft.

This language system consists of two programs – an editor for creating PILOT listings, and a translator that converts the PILOT program to Applesoft and appends the necessary BASIC utilities needed to make everything work properly. To interface the plotter to the language, one needs only to modify two subroutines and add one new subroutine to the translator program. To keep these programs clear, I will show only the changes that are to be made in the program published in Poole's original article. If you used different line numbers in your version, in order to see where you should put them you will have to compare these changes with the original listing.

The modifications to the translator perform three tasks:

1. We must load the plotter driver routine and initialize the system. Since the routine starting at line 50000 is used at the beginning of every translated program, this is where we will add these tasks.

2. We must add plotter commands after the screen drawing commands so that our plotted image will appear at the same time it is being drawn on the screen. The screen drawing routine begins at line 55000, so this is where we will make these changes.

3. Finally, we need to add a routine that scales the plot commands for the paper size and plotter resolution, sets the pen in the up or down position as appropriate, and ships this assemblage of data to the plotter for execution. We will create this routine starting at line 56000.

Because all the changes are in that portion of the translator appended to each translated program, only one tiny change needs to be made in the PILOT programs themselves. As implemented, the command G:GOTO x,y will only be executed when the next G:DRAW command is given. If you are moving the turtle to a new loction X,Y with the pen up, you can execute this on the plotter with the sequence:

```
G:PEN UP;GOTO X,Y;DRAW 0;PEN DOWN
```

The function of the DRAW 0 command is to force the plotter to carry this motion out before setting the pen down.

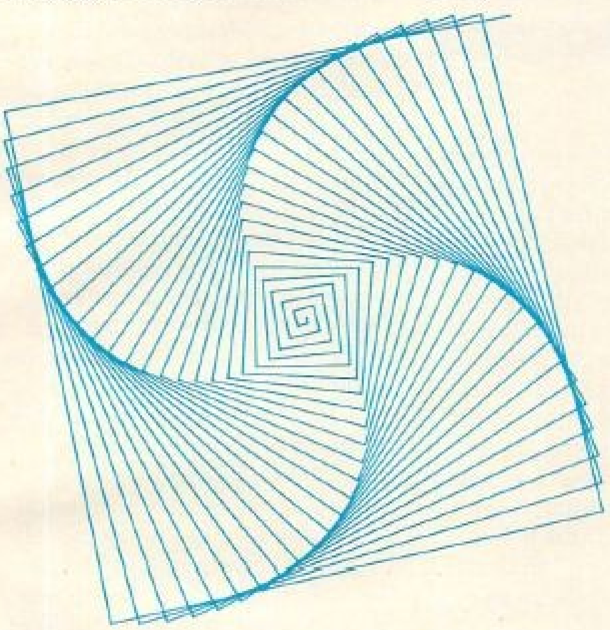
Except for the small inconvenience of adding the extra DRAW 0 commands after each GOTO, any of your existing PILOT turtle graphics programs will run on the plotter as soon as they have been re-translated. I recommend using the original translator for making sure the picture fits on the screen and otherwise does what you want. Once this is done, you can use the modified translator (called, for example, TRANSPLOT) to generate the BASIC program that will both draw pictures on the display and plot them on the plotter at the same time.

To try out the plotter, I entered the following PILOT program:

```
*SQUIRAL
```

```
G: CREAL
C:A=0
*LABEL
G:DRAW A
G:TURN 91
C:A=A+1
J(A<100):*LABEL
E:
```

When this was translated and run, I was able to get a beautiful squiral pattern that was devoid of the jaggies one gets with a raster display screen.

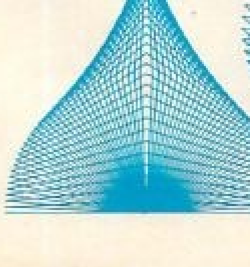


As you create pictures of your own, you will want to change pen colors every so often in the middle of a drawing. An easy way to do this is to use the following procedure when you want to change colors:

```
*QUERY
T:CHANGE PEN AND PRESS RETURN
A:
E:
```

This will stop the execution of the program while you change pens. When you are ready to start plotting again, just press RETURN.

The following figures are but a small indication of the pleasures that await those of you who want to increase the resolution of your turtle graphics.



The changes to be made in the *Translator* program of Apple Turtle PILOT include:

1. Set up procedure:

```
50000 PRINT CHR$(4);"BLOAD PLOT1.8"
50002 HIMEM: 35071
50004 CALL 35081
50006 DIM Q$(25),Q$(31)
50008 QP=1:QX=0:QY=0:GOSUB 56000:QP=0
```

2. Modify drawing routine:

```
55004 GOSUB 56000
55045 IF QP=1 THEN GOSUB 56000
55060 HPLOT TO QX+139.0005, -QY+80.0005:
      GOSUB 56000
55070 RETURN
```

3. Add plotter routine:

```
56000 XI=20*(QX+137.5):YI=20*(QY+106.25):
      P%=QP+2
56010 IF XI<0 THEN XI=XI+65536
56020 IX%=XI/256
56030 POKE 35085,IX%
56040 IX%=XI-IX%*256
56050 POKE 35084,IX%
56060 IF YI<0 THEN YI=YI+65536
56070 IY%=YI/256
56080 POKE 35087,IY%
56090 IY%=YI-IY%*256
56100 POKE 35086,IY%
56110 IF P%<0 THEN P%=P%+255
56120 POKE 35088,P%
56130 CALL 35072
56140 RETURN
```

Note: If you are extremely picky about plotting accuracy, add the line:

```
56005 XI=XI*1.0007506:YI=YI*1.0198781
```

Any disk that contains programs generated with *Transplot* also needs to have a copy of the Strobe program *Plot*1.8. To copy this program to your disk, place any Strobe disk in your Apple and enter:

```
BLOAD PLOT1.8
```

Next, insert your program disk and enter:

```
BSAVE PLOT1.8,A$8900,L$6E0
```

©

***IBM-PC** ****TRS 80-II** *****ATARI**

"Quality Computer Products at Sensible Prices"

EARTH ATTACKERS • Great graphics, quick action • TRS 80-II, with Basic	**19.95**
UTILITIES • Collection of software tools (Printer setup, file conversion and more) • IBM-PC with DOS Basic	**19.95**
MAIL LIST and LABEL control program • User Friendly • IBM-PC	**49.95**
BASIC CROSS REFERENCE • TRS 80-II with Basic	**49.95**
ATARI-400 48K upgrade kit (introductory offer) • Simple 1-hour installation • Everything you need (except soldering iron)	**99.95**

MSX **Micro Systems Exchange**
P.O. Box 4033
Concord, CA 94524
(415) 355-7130

*Trademark International Business Machines Corp. **Trademark Tandy Corp. ***Trademark of Atari, Inc.

Apple Educational Games

Sheila Cory, Chapel Hill, NC

If you are either a teacher or a parent of young children and have access to an Apple II+ computer with 48K and a disk drive, there is some software available that you should know about. Produced by The Learning Company, it's specifically designed for preschool and elementary-school youngsters.

This review covers three packages of programs. The first, *Juggles' Rainbow*, is designed for children aged three to six. The second, *Bumble Games*, is for ages four to ten, and the third, *Bumble Plot*, eight to thirteen. All three packages are well designed, and the sequence of the material progresses logically.

Juggles' Rainbow

Juggles' Rainbow is a welcome addition to the small amount of good software for the preschool, kindergarten, and first grade set. Frequently, teachers of very young children are left out when computers are discussed in faculty meetings or workshops, and feel that there's not much that can be done with the computer for children who don't yet have reading skills. It takes great sensitivity to the particular qualities of children of this age to produce software that is interesting, challenging without being too difficult, and educationally sound. *Juggles' Rainbow* shows this sensitivity.

Juggles' Rainbow consists of three programs for children, and one program for teachers or parents. The children's programs are Juggles' Rainbow (the name is used for the entire package and for one of the programs within the package), Juggles' Butterfly, and Juggles' Windmill. The adults' program, called The Big Question Mark, allows the setting of options such as whether sound should be included in the program, whether the child should be given picture and word or just word clues, and gives instructions for dividing the keyboard into halves and quarters for some of the exercises.

Juggles' Rainbow is designed to reinforce the teaching of the concepts of *above* and *below*. The program divides the keyboard into an upper and a lower section with a blue strip of cardboard that is provided with the diskette. A blue line appears on the screen. Children find that when they depress a key below the keyboard divider, a colorful vertical line appears below the blue screen line, and when a key is depressed above the keyboard divider, a colorful line appears above the screen line.

The next segment of this program prompts the user to depress keys above and below the keyboard divider to color in outlined bars above and below the blue screen line. The third segment allows the child to apply his skill with *above* and *below* to create a colorful rainbow.

Juggles' Butterfly reinforces the concepts of *left* and *right*. Again the keyboard is divided with a provided blue strip, but this time the division is in a vertical direction, creating a left and right section of the keyboard. The program works basically the same way as Juggles' Rainbow, but the final segment allows the child to create a marvelous butterfly by applying color to the right and left sides of the butterfly body as keys to the right or left of the keyboard divider are depressed.

Juggles' Windmill takes the learning one step further by having the child depress keys above (or below) the horizontal keyboard divider and to the left (or right) of the vertical divider. The culmination of this activity is the creation of a windmill that would delight a very young child.

Luring our four-year-old visitor, Christopher, away from his LEGO project to try out these programs was difficult, but they quickly absorbed him. This was not only his first opportunity to use the programs, but was also his first time using a computer. A good deal of adult guidance was needed to help him figure out what he was supposed to do, and to extend the learning. This program could make ideal use of a classroom volunteer or older child whose role would be to talk through the concepts, exclaim over the results, and guide the discoveries made using the computer.

One problem Christopher had was keeping the cardboard keyboard dividers in place. I recommend that a piece of heavy blue yarn be used instead of the cardboard. The yarn could be placed between the second and third rows of the keyboard, rather than over the third row as suggested in the manual, and the yarn could be securely taped in place at each end. A similar procedure could be used for the vertical keyboard divider. Christopher's interest in the activity lasted about ten minutes, giving him time to get through Juggles' Rainbow and begin Juggles' Butterfly. His enjoyment of the activity was evident when he asked me if he could play the rainbow game again before he went home.

Bumble Games

Bumble Games introduces the delightful Bumble, who is the central character in all of the programs in *Bumble Games* and *Bumble Plot*. The learning objective in *Bumble Games* is to teach the graphing of positive numbers. Some of the concepts covered are also covered in the MECC (Minnesota Educational Computing Consortium) game of Hurkle. The *Bumble Games* diskette contains six programs, each one progressively more sophisticated. The sequence is excellent, extending the learning by a small degree with each successive game.

The program Find Your Number begins the sequence by giving practice in finding a number between zero and five that Bumble has secretly chosen. The child is shown a horizontal or a vertical number line with the numbers zero to five on it, and makes a guess. Bumble responds with a left or right arrow in the case of a horizontal number line, and an up or down arrow in the case of a vertical number line, indicating whether the next number guessed should be more or less than the present guess. The horizontal and vertical number lines begin preparing children for an X and Y axis that they'll see in a later program. When the number is guessed, the child gets a colorful display of the number, accompanied by tones representing the number. If two children want to play this game, Bumble will select two numbers.

Find the Bumble introduces a four-by-four grid, cleverly differentiating the X and Y axes by labeling one with letters and one with numbers. Bumble hides in one of the boxes formed by the grid, and the child must find Bumble by naming the coordinates of his box. Bumble is very helpful, telling the child to pick bigger or smaller numbers for the Y axis, and letters to the left or to the right for the X axis. When Bumble is found, his friendly, bigger-than-life image appears on the screen.

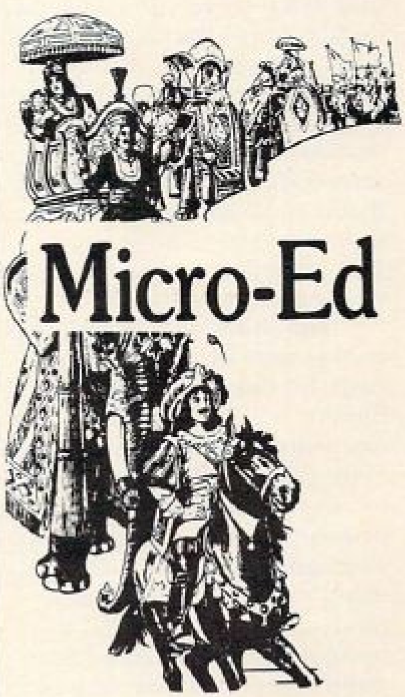
Butterfly Hunt has Bumble out searching for his lost butterfly. This game works very much like Find the Bumble, but a slightly larger grid prepares the child for the next game, Visit From Space.

Visit From Space introduces the idea that the intersection of two lines in a grid can be named by using a number on the bottom of the grid and one at the side. In this game, both X and Y axes are labeled with numbers. The object of the game is for the child to find Bumble's cousin who has flown in from outer space and is hiding in his spaceship somewhere on the grid. Very clear graphic and written clues help the child learn to locate exactly the intersection he wants to guess. When the spaceship is finally found, it zooms across the screen, making appropriate outer-space noises!

Tic Tac Toc is a game for two players, similar to the more conventional tic tac toe. The idea is for the child to get four markers in a horizontal, vertical, or diagonal line before his or her opponent does. The game screen consists of a five-by-five grid, and a marker is placed by naming the coordinates of the desired position.

The board is somewhat con-

Join the parade to



Micro-Ed

educational software

Send for free catalogs
Specify: Pet • VIC • TI • Apple
• Commodore 64
• TRS-80 Color Computer
• TRS-80 Model III
• Atari

telephone us at
612-926-2292

Micro-Ed Inc.
P.O. Box 24156
Minneapolis, MN 55424

fusing to the beginning player; it would be helpful for the teacher to make a similar board on a transparency and use the overhead projector to play the game a few times with the whole class before children begin to play the game on the computer. The game does give excellent practice in naming points on a grid. It is just different enough from tic tac toe to be interesting.

Bumble Dots extends the grid to ten-by-ten. In this game, Bumble helps the child draw dot-to-dot pictures. A dot appears on the grid, and the child is asked to name it. When the first dot is successfully named, a second dot appears, and when that is successfully named, a line is drawn to connect the dots. This procedure continues until a whole picture is drawn.

The child can also make his own picture by naming coordinates for Bumble to connect. Bumble first asks the child how many dots will be in his picture. Since this is difficult for a child to ascertain in advance, it would be helpful if the teacher had the children first draw a picture using three to nine dots on a piece of graph paper, and then bring that picture to the computer when their turn comes. Children would then be all set to answer Bumble's question about the number of dots needed for the picture. The Learning Company, in developing this program, recognized the fact that generations of children have loved dot-to-dot pictures, and that a natural progression of learning can take place by tapping into this love.

Bumble Plot

Bumble Plot extends the learning about grids to include negative numbers. It consists of five programs, again carefully sequenced to take the child comfortably through the steps culminating with naming points on a ten-by-ten grid where negative numbers are used and the 0,0 point is in the middle. The sequence starts with Trap and Guess, where the child tries to trap Bumble's secret number on a minus three to plus three horizontal or vertical number line. Bumblebug has Bumble hopping around on a grid; the object is to set traps for him to jump into! In Hidden Treasure, the child searches for invisible treasures on a ten-by-ten grid with negative numbers. I found a ship's anchor, a diamond ring, a friendly octopus, and a golden crown! Children would enjoy a worksheet where they could show what they found and where they found it when they played the game. These worksheets could be displayed on the bulletin board above the computer.

Bumble Art is similar to Bumble Dots, but contains negative numbers in the grid. The most action-packed game of the series is Roadblock. The object of this game is to build roadblocks to surround the bank robber before he gets away. This, of course, all takes place on a minus-five by plus-five grid, providing wonderful practice in the skills that have been developed through all of the other games.

All three of the packages reviewed here share some very positive qualities. They all contain excellent graphics; they use sound appropriately to enhance the learning or entertainment value of the program, and sound can be turned off if it provides a distraction in the classroom. The programs are very user friendly, take all kinds of input without bombing, give the user excellent prompts, and have very carefully formatted screen displays.

Manuals are well illustrated and appealing. Each one gives instructions on how to load the diskette and a little information about each program. It would have been very useful to have included suggestions for teachers about things to talk about before each program, and appropriate worksheets for follow-up activities.

The company will send you a set of activity cards for free when you send back the owner registration card. This card also entitles you to purchase a backup diskette for $12. No teacher should ever use software in the classroom without having a backup diskette.

I suspect that schools that purchase software from The Learning Company will have a new little character joining Snoopy and The Cat In The Hat in adorning their bulletin boards. Bumble has great personal appeal and represents software that is both educationally sound and fun to use.

Juggles' Rainbow ($45)
Bumble Games ($60)
Bumble Plot ($60)
The Learning Co.
4370 Alpine Road
Portola Valley, CA 94025 ©

PROMQUEEN (VIC-20 Hardware)

Harvey B. Herman, Associate Editor

The hardware reviewed here will be of interest to a select group of **COMPUTE!** readers. If you own a VIC and have the need to "burn EPROMs," you should consider this cartridge. On the other hand, if you are completely befuddled by the previous sentence, go to the next article; save $200.

I was excited when I received the PROMQUEEN for review, as it was just what I needed. Several pieces of computer-related equipment which I use daily contain EPROMs (Erasable Programmable Read Only Memory chips). What would I do if one failed and I had no way to replace it? The PROMQUEEN promised to solve this potential problem, even for one like myself, who had never programmed an EPROM before.

It is misleading to think of the PROMQUEEN exclusively as

Apple Machine Language Memory Aid

K. Lourash, Decatur, IL

"ML Helper" is a utility developed to assist fledgling Apple machine language programmers in studying 6502 object code when the original source code is not available, and also in adapting that code to their particular needs and systems. This program also works as is on OSI and can easily be modified for any Microsoft BASIC.

Options are offered in this program to list and modify zero page usage, to list and modify absolute addressing references, and to relocate the code under examination. Although written in Microsoft floating-point BASIC, this utility is readily converted to the other popular dialects. In fact, while my system is OSI, the listing is for Apple simply to involve a wider audience.

You may save ML Helper without REMarks. If you do, notice that line 31 may be incorporated into line 29, and line 35 into line 33, for increased program optimization. However, do not tamper with the "NEXT A" statement of line 51, since ML Helper will exit a loop without completing it; a simple "NEXT" there is insufficient.

In the interest of brevity, I chose to do no error checking of input from the keyboard. Thus it's easy to become careless and obtain seemingly inexplicable program performance. Also, when using hexadecimal notation, I assumed you won't prefix an address with the "$" symbol. Furthermore, leading zeros are harmless, but quite unnecessary. No relocate is foolproof. Hence, ML Helper does not resolve the indirect JMP or the technique of jumping with an RTS once the stack has been prepared. In other words, jump tables and data blocks are moved unchanged.

Disassemble And Relocate

When up and running, ML Helper emulates a disassembler, examining the address range you've specified for valid 6502 operation codes. When they are found, the program logic proceeds to list or modify the zero page references, to list external absolute references, to modify absolute references, or to move code and modify addresses for a successful relocate, whichever option is operative.

Bytes determined to be invalid instruction code sequences are assumed to form data tables. A data table finder, as such, is always active and can actually become an unspecified sixth option to locate unknown data table areas.

At this point I set an arbitrary criterion – namely, that wherever there occurs a block of six or fewer consecutive bytes of executable code, the data table finder should, nonetheless, report that block of code as part of a data table area. If this standard proves unsuitable for your requirements, then change the "A-7" expression in line 350. The absence of data tables is reported as an address range of 0-0 ($0000-0000 hexadecimal).

Menu item 4 may not be immediately clear. The "EXTERNAL" references that ML Helper will list are those absolute addresses referencing memory outside the body of the program module being examined. Displaying all absolute addressing usage produces a counterproductive volume of screen clutter which I thought best to avoid.

Menu item 5, by which you elect to change absolute references, is not similarly restricted. If during a run it appears that interesting data might scroll away, then Apple users are reminded to invoke the CTRL S Stop-List feature of their system; others may have to rely on CTRL C or divert all output to hard copy. Have fun exploring uncharted machine language programs with ML Helper pointing the way.

```
0 DATA 232,200,202,136,72,104,24,56,96,170,1
    68,138,152,234,10,74,42,106,186
1 DATA 154,64,120,88,184,248,216,8,40,0,208,
    240,144,176,48,16,80,112,169,162
2 DATA 160,201,224,192,105,233,41,9,73,165,1
    66,164,133,134,132,230,198,197,228
3 DATA 196,101,229,36,37,5
10 DATA 69,38,102,6,70,181,182,180,149,150,14
    8,246,214,213,117
11 DATA 245,53,21,85,54,118,22,86,177,145,209
    ,113,241,49,17,81,161,129,193,97
12 DATA 225,33,1,65,32,76,108,44,173,174,172,
    141,142,140,238,206,205,236,204
13 DATA 109,237,45,13,77,46,110,14,78,189,190
20 DATA 188,157,254,222,221,125,253,61,29,93,
    62,126,30,94,185,153,217,121,249
21 DATA 57,25,89: GOTO 530
30 REM *** LIST ADDRESSES ***
40 IF A(Z) >  = S THEN  IF A(Z) <  = E THEN
    RETURN
50 IF Z = 0 GOTO 80
60 FOR X = 0 TO Z - 1: IF A(X) = A(Z) THEN  R
    ETURN
68 S(T) = VAL (H$):E(T) = VAL (E$)
70 NEXT
80 PRINT "ADDR REF'D:  ";: IF H THEN D = A(Z)
    : GOSUB 220: PRINT "$"H$: GOTO 100
90 PRINT A(Z)
```

```
100 Z = Z + 1 - (Z > 29): RETURN
110 REM *** ZERO PAGE CHANGE ***
120 FOR I = 0 TO X: IF C(I) = A(Z) THEN POKE A
    + 1,D(I)
130 NEXT : RETURN
140 REM *** RELOCATE ***
150 IF A(Z) < TS OR A(Z) > TE THEN  RETURN
160 I =  PEEK (A + 1) + T3: IF I > 255 THEN I
    = I - N:T4 = T4 + 1
170 POKE A + 1,I: POKE A + 2, PEEK (A + 2) + T
    4: RETURN
180 REM *** CHANGE ABSOLUTE ADDR ***
190 FOR I = 0 TO X: IF C(I) = A(Z) THEN K =  I
    NT (D(I) / N)
195 POKE A + 1, D(I) - N * K: POKE A + 2,K
200 NEXT : RETURN
210 REM *** DEC-HEX ***
220 H$ = "":F = 4096: FOR J = H TO 4:K =  INT
    (D / F):D = D - K * F
225 H$ = H$ + MID$ (G$,K + H,H):F = F / 16: NE
    XT : RETURN
230 REM *** HEX-DEC ***
240 D = 0:F = H: FOR J = LEN (H$) TO H STEP -
    H:M = ASC ( MID$ (H$,J,H )) - 48
245 D = D + F * (M - 7 * (M > 9)):F = 16 * F:
    NEXT : RETURN
250 REM *** PRINT DATA TABLES ***
260 PRINT "DATA TABLE:  ";: IF H THEN D = T1:
    GOSUB 220: PRINT "$"H$"-";:D=T2
265 GOSUB 220: PRINT H$: RETURN
270 PRINT T1"-"T2: RETURN
280 REM *** MAIN ROUTINE ***
290 FOR A = S TO E
300 REM *** SKIP DATA TABLES ***
310 FOR I = 0 TO T: IF S(I) THEN IF A > = S(I)
     THEN A = E(I) + 1:S(I) = 0
320 NEXT : FOR I = 0 TO 150: READ M: IF PEEK (
    A) = M GOTO 390
330 NEXT
340 REM *** PRINT DATA TABLES ***
350 IF A - 7 > T2 THEN IF T1 THEN GOSUB 260:T1
     = A
360 IF T1 = 0 THEN T1 = A
370 T2 = A: GOTO 510
380 REM *** 1-BYTE IGNORE ***
390 IF I < 29 GOTO 510
400 REM *** 2-BYTE IGNORE ***
410 IF I < 48 GOTO 500
420 REM *** ZERO PAGE ***
430 IF I > 102 OR C > 2 GOTO 470
440 IF C < 3 THEN A(Z) = PEEK (A + 1): ON C GO
    SUB 50,120
450 GOTO 500
460 REM *** 3-BYTE ***
470 IF I < 103 GOTO 500
480 IF C > 2 THEN A(Z)=PEEK (A + 1) + PEEK (A
    + 2) * N: ON C-2 GOSUB 150,40,190
490 A = A + 1
500 A = A + 1
510 RESTORE : NEXT A: GOSUB 260: END
520 REM *** END OF MAIN ROUTINE ***
530 PRINT "1= LIST ZERO PAGE REFERENCES":PRINT
    "2= CHANGE ZERO PAGE REFERENCES"
531 PRINT "3= RELOCATE": PRINT "4= LIST EXTERN
    AL ABSOLUTE REFERENCES"
532 PRINT "5= CHANGE ABSOLUTE REFERENCES": PRI
    NT: PRINT "CHOOSE ONE: ";: GET H$
533 PRINT H$:C = VAL (H$):PRINT :PRINT "WANT H
    EX  NUMBERS, Y/N? ";: GET H$:PRINT H$
540 PRINT :H = H$ = "Y":N = 256:G$ = "01234567
    89ABCDEF": DIM A(30)
541 INPUT "INPUT START,END ADDRESSES: ";H$,E$:
     PRINT
542 IF H THEN GOSUB 240:S = D:H$ = E$: GOSUB 2
    40:E = D: GOTO 560
550 S = VAL (H$):E = VAL (E$)
560 IF C < > 3 GOTO 660
570 INPUT "INPUT TARGET ADDRESS: ";H$: PRINT :
     IF H THEN GOSUB 240:TS=D:GOTO600
580 TS = VAL (H$)
590 REM *** CALCULATE OFFSET ***
600 TE = TS + E - S:I = ABS (TS - S):T4 = INT
    (I / N):T3 = I - T4 * N
605 IF TS < S THEN T3 = - T3: T4 = - T4
610 REM *** MOVE ROUTINE ***
620 IF T3 > 0 THEN K = TE: FOR I = E TO S STEP
     - 1: POKE K, PEEK (I):K=K-1
625 NEXT : GOTO 650
630 K = TS: FOR I = S TO E: POKE K, PEEK (I):
    K = K + 1: NEXT
640 REM *** SWAP TS & S, TE & E ***
650 K = TS:TS = S:S = K:K = TE:TE = E:E = K
660 PRINT "LIST UP TO 11 KNOWN DATA TABLES IN
    THE  PROGRAM.  TYPE 0,0 WHEN DONE.":P
    RINT
670 PRINT "DATA TABLE "T" START,END:  ";: INPU
    T "";H$,E$
675 IF H THEN GOSUB 240:S(T) = D:H$ = E$: GOSU
    B 240:E(T) = D: GOTO 690
680 S(T) = VAL (H$):E(T) = VAL (E$)
690 IF E(T) THEN I = T3 + T4 * N:S(T) = S(T) +
     I:E(T) = E(T) + I:T = T + 1
695 IF T < 11 GOTO 670
700 IF C < > 2 THEN IF C < > 5 THEN PRINT : GO
    TO 290
710 PRINT :PRINT "LIST UP TO 11 ADDRESSES TO B
    E CHANGED. TYPE 0,0 WHEN DONE.":PRINT
720 PRINT "#"X". OLD,NEW ADDRESSES:  ";: INPUT
     "";H$,E$
725 IF H THEN GOSUB 240:C(X) = D:H$ = E$: GOSU
    B 240:D(X) = D: GOTO 740
730 C(X) = VAL (H$):D(X) = VAL (E$)
740 IF C(X) = D(X) OR X = 10 THEN PRINT : GOTO
     290
750 X = X + 1: GOTO 720
```

TIRED OF WRITING BORING BUSINESS DATA PROCESSING PROGRAMS?

THE ALIEN GROUP needs an imaginative Programmer to create interesting software for its Voice and Music Synthesizer Peripherals. He/she must be capable of writing in Assembly Language and have detailed knowledge of at least one of the popular home computers and its operating system. Send Resume to: THE ALIEN GROUP, 27 W. 23rd St., NY, NY 10010

Use the handy reader service cards in the back of the magazine for information on products advertised in **COMPUTE!**

Keith Falkner

Tap Applesoft's Heartbeat

You can use machine language routines to enable Applesoft to read and rapidly process incoming data.

Imagine that your Apple is connected to some gizmo which feeds the Apple some data rapidly. The device could be, for example, a modem or some newfangled digital geiger counter monitoring an atomic reactor. In an example below, we will simulate this device with the game paddle buttons, or, if you have no paddles, with a mere piece of wire. The essential idea is that the attached device offers data to the Apple sporadically, and the data will be lost if it is not noticed and processed within a few milliseconds.

If you try to support this device with a program written in Applesoft BASIC, you will likely miss some of the data offered by the device, because Applesoft is rather slow. Assuming that such a problem does arise and must be solved, here's how.

Machine Language Patch Into CHRGET

Here is an intriguing exercise: type in and run the listing in Program 1. If you type it correctly, it will say "OK"; make sure you fix it if it says "OOPS." This program installs, but does not run, three tiny machine language routines. Now type CALL 909 and then run the program again. Inexplicably, it will make an irritating buzz for the 0.37 seconds it takes to run. Indeed, you can load and run almost any Applesoft program and listen to it run.

You may notice that difficult computations and lengthy array references are accompanied by buzzes, whereas fast-running code such as FOR/NEXT loops that do little more than count will produce brief musical tones. I do not suggest that this is a useful effect, but I hope it sparks your interest, for what is coming is a bit dull and difficult but results in a very powerful technique which you can harness to produce utterly amazing results at zero cost.

By the way, you can deactivate the noise-making routine and restore your Apple to normal by typing CALL 896. The DOS command FP is even more powerful; issue that if your Apple seems confused.

A Look Into CHRGET

Here is how the noise is caused. The Applesoft interpreter uses a tiny routine to fetch each byte of your program in turn as the program runs. The (valid) BASIC statement IF BAD THEN STOP is stored as six bytes, specifically the token for IF, the letters B, A, and D, and the tokens for THEN and STOP. The character-getting routine, which is known by the name CHRGET, will be invoked a total of seven times to execute all of this statement (the token for THEN is fetched twice, once to detect the end of the variable name BAD, and once to be executed).

Program 1 and the routine installed at location 909 introduce a detour into CHRGET so that the Apple's speaker is tweaked each time a character of the program is fetched. This of course makes the noise and accounts for the various buzzes and squeaks made by slow- and fast-running code. To see the actual machine language routine, enter the monitor via CALL -151 and enter 380L (number 380 followed by letter L) to see the routines at 896 ($0380), 909 ($038D), and 922 ($039A).

The CHRGET routine starts in location 177 ($00B1), and can be listed by B1L (letter B, digit 1, letter L). You can verify if you wish that CALL 909 installs a JMP instruction at location 186 ($00BA), and CALL 896 restores the CMP and BCS instructions which belong there. You can return from the monitor to Applesoft by typing CTRL-C and pressing RETURN.

Now let's put this technique to use. If you have game paddles, identify PDL (1) and skip the rest of this paragraph. To simulate the button on PDL(1), you will need a piece of slender wire at least two feet long. Solid wire works better than multi-strand. Strip about one-eighth inch from each end. You shuld now *turn off* the Apple and open the cover carefully. Locate the GAME I/O connector at coordinates J8 on the motherboard, and stick an end of the wire into hole number three, which is third from the front on the right side.

Do be careful with this, because disaster awaits you if you pick the wrong hole, or are careless with the other end of the wire. Now close the cover of the Apple, letting the free end of the wire hang down away from the computer. Reach under the front edge of the keyboard and you will find the heads of some bolts. You will be touching the free end of that wire to one of these to simulate a press of the button. If you choose, you can loosen one of these, attach another piece of wire, tighten the bolt, and attach the two loose ends of wire to any type of switch, but this is not essential. When these preparations are complete, turn the Apple on again.

Catching Every Count

Now type in Program 2 and run it. Please note the lengthy loop in lines 130-140. This takes over half a minute to run and obviously contains none of the PEEK statements necessary to test for a press of button number one. Those tests are done by the machine language routine patched into CHRGET, at locations 922 through 965. When the program is running, press the button (or touch the wire to the bolt) as fast as you can count, and you will find that the Apple catches every single one. Actually, when you try to touch the wire to the bolt once, you almost certainly cause it to bounce and touch the bolt more than once, so the count will be higher than you expect, and never lower.

In this example the switch was tested by a few instructions in machine language. This powerful technique is possible only in machine language. Perhaps it is possible to devise a routine that would permit a few lines of BASIC to be invoked by the routine which interrupts CHRGET, but what would be the point? Our objective here is to support a rapid-fire device, and any attempt to do this in BASIC will, it is assumed, lead to missed data. At least that is where this article started.

Using The Keyboard Buffer

A totally practical application of intercepting CHRGET is a keyboard buffer, except for one troublesome detail. From time to time, in any program which handles strings, Applesoft must pause to accomplish "garbage collection" – in other words, to make available again some memory which has been used for storage of strings which were later discarded. This process usually takes from one to thirty seconds, but in an artificial and extreme case it could take over an hour!

During "garbage collect," Applesoft is totally out of touch with all external events, so the keyboard buffering routine has no way to service the keyboard. Nonetheless, the routine is of genuine help when a speedy typist is using a slow data entry program. In fact, even a moderately slow hunt-and-peck typist like me can occasionally leave Applesoft behind. With the buffer running, I never lose a key.

There are two other limitations. During processing of the LIST command, Applesoft is not using CHRGET, so the buffering routine has no chance at the keyboard. Also, when DOS is active, all BASIC functions are inactive, so again the keyboard cannot be serviced.

Program 3 shows the complete keyboard buffer program. The program occupies the first 512 points of the BASIC program area, so it destroys any Applesoft program already present.

Briefly, here is how the program works. A preliminary test verifies that Applesoft is active, for this program is inapplicable to Integer BASIC. Next, the program sees if the beginning-of-BASIC pointer has been altered to $0A01 (from the usual $0801). If so, a warm start is done, retaining the current Applesoft program; if not, the pointer is so altered, and the new routine of Applesoft is called. Then the "patch" to CHRGET is made, as in Programs 1 and 2.

The next step is a connection to the keyboard-servicing routine at the "hook" known as KSW. Whenever such a connection is what you need, you must let DOS know your intentions, or it will patiently remove your connection and restore its own hook. This is very easy – just CALL 1002 (or JSR $3EA in machine language). The program ends by entering Applesoft at the warm-start entry $E003.

By this point, the program really has not done anything except insinuate itself into the system and protect itself from harm. The actual buffer is the 256-byte area from $0900 to $09FF (2304 to 2559), and two one-byte counters look after data in the buffer. The counter BIX points to the next place where a key can be stored, and the counter BOX points to the next byte to be sent to whoever asks for a key.

For example, if BIX contains $2E and BOX contains $28, the operator has keyed six bytes ahead, and they are stored in locations $0928 through $092D. If the operator now keys exactly 250 more bytes before the running program asks for any more, the keyed bytes will be stored in $092E through $09FF, then the buffer will "wrap around" and more keys will be stored in $0900 through $0926. By this time the value in BIX will be $27, one less than that in BOX. That's 249 in addition to the six already there, and now the buffer is full, so the buffering routine will sound the "bell" when it cannot store the last byte keyed.

At this point the operator must pause and wait for the program to catch up. I think this event is very unlikely.

Keys are detected and stored by the routine patched into CHRGET. A word of caution to anyone patching CHRGET: since BASIC uses this routine dozens or thousands of times a second, the patch must execute as fast as possible, else the program may be slowed to an unacceptable degree.

Does It Function?

When a key is wanted, the code at INLINK sees if one is in the buffer. If not, the standard ROM routine is called. If a key is available in the buffer, it is delivered, and the counter, BOX, is updated to account for the departed key. It is all very simple, mainly because of the eight-bit indexing automatically provided by the 6502's X-register. Indeed, if the buffer were any size but 256 bytes, the program would have been noticeably harder to write and debug.

OK, how do you key this program into your Apple? You could CALL -151 to get to the monitor, then type in all the hex stuff, 803:4C 09 08 4C 99 08, and so on. If you did the "homework" I assigned in last month's column, there is an easier way. Key in the pure Applesoft program in Program 4, then SAVE it, RUN it, and finally EXEC GEN KEYBUF. This final step will invoke the mini-assembler to build KEYBUF, save the result, and return control to the keyboard eventually. This process must destroy any Applesoft program in memory, so be sure you have saved Program 4 before typing the EXEC command!

To verify all this work, peer closely at the screen – the command JMP $083C should be in location 08A8. The acid test, of course, is "does it work?" Follow the instructions below to test your work, and when you actually make it work, you'll have a potent and versatile tool which makes your Apple a little bit better than it was before!

Homework Assignment. Boot your System Master and LOAD BRIAN'S THEME. That is the program which displays pretty moiré patterns in high resolution. Here is some code to add a fascinating effect! Type in the few lines in Program 5 and RUN the changed program. When the display starts acting oddly, play with the keyboard. The most recently pressed key controls the timing in a tiny machine language routine at location 600 ($258).

In my particular Apple, the keys W, K, 8, question mark, and especially CTRL-D, produce interesting effects. The machine language routine is completely relocatable, so it can be used without change in any place in memory where 26 bytes are free. So if you wish to use the routine in another program, change the variable ML to whatever suits you. The timing is so delicate that the effects change greatly when ML is just under a multiple of 256, so that a branch instruction crosses a page boundary. To stop this demonstration, you must press RESET, because the machine language routine treats CTRL-C as any other key.

Table: How to use the Keyboard Buffer

1. To load and initialize the routine,
   BRUN KEYBUF
2. Now use your Apple as usual, but be sure that you do not switch to Integer BASIC!
3. To suspend use of the buffer,
   CALL 2054
4. To resume use of the buffer,
   CALL 2051
5. To recover memory used by the buffer, after suspending it via CALL 2054,
   FP (or INT, if you choose)
6. To copy the routine from disk to disk,
   BLOAD KEYBUF
   Insert the disk to receive a copy.
   BSAVE KEYBUF,A$803,L$F8

Program 1.

```
10 REM   'TAP' DEMO 1
20 FOR I = 896 TO 935
30 READ X
40 Z = Z + X
50 POKE I,X
60 NEXT
70 IF Z <  > 5155 GOTO 90
80 PRINT "OK": END
90 PRINT "OOPS. Z=";Z: END
896 DATA 169,201,133,186,169,58
902 DATA 133,187,169,176,133,188
908 DATA 96, 169,76,133,186,169
914 DATA 154,133,187,169,3,133
920 DATA 188,96,141,48,192,201
926 DATA 58,176,3,76,190,0
932 DATA 76,200,0,0
```

Program 2.

```
10 REM   'TAP' DEMO 2
20 REM
30 FOR I = 896 TO 955
40 READ X
50 Z = Z + X
60 POKE I,X
70 NEXT
80 IF Z < > 7425 THEN PRINT "OOPS. Z=";Z: STOF
90 HOME : GR
100 PRINT "WHILE I SCRIBBLE AIMLESSLY,"
110 PRINT "PRESS BUTTON 1 SEVERAL TIMES."
120 POKE 24,0: POKE 25,0: POKE 26,0: CALL 909
130 FOR I = 1 TO 1000: COLOR= 16 * RND (I)
140 PLOT 40 * RND (I),40 * RND (I): NEXT
150 CALL 896:T =  PEEK (25) + 256 * PEEK (26)
160 TEXT : PRINT CHR$ (7): REM BELL!
170 HOME : PRINT "YOU PRESSED IT "; INT (T / 2
    );" TIMES."
```

```
896 DATA 169,2Ø1,133,186,169,58
9Ø2 DATA 133,187,169,176,133,188
9Ø8 DATA 96, 169,76,133,186,169
914 DATA 154,133,187,169,3,133
92Ø DATA 188,96,72,152,72,173
926 DATA 98,192,41,128,197,24
932 DATA 24Ø,8,133,24,23Ø,25
938 DATA 2Ø8,2,23Ø,26,1Ø4,168
944 DATA 1Ø4,2Ø1,58,176,3,76
95Ø DATA 19Ø,Ø,76,2ØØ,Ø,Ø
```

Program 3.

```
ØØØØ               ; THIS PROGRAM USES 512 BYTES FROM
ØØØØ               ; 2Ø48 TO 2559 TO CONTAIN AND LOOK
ØØØØ               ; AFTER A 256-BYTE KEYBOARD BUFFER.
ØØØØ               ;
ØØØØ               ; 'BRUN KEYBUF' TO CREATE THE BUFFER.
ØØØØ               ; 'CALL 2Ø54' TO DISABLE THE BUFFER.
ØØØØ               ; 'CALL 2Ø51' TO RE-ENABLE THE BUFFER.
ØØØØ               ;
ØØØØ               ; HOW TO SAVE THE PROGRAM:
ØØØØ               ; BSAVE KEYBUF,A$8Ø3,L$F8
ØØØØ               ;
ØØØØ                      *=$8Ø3          ;START AT 2Ø51.
Ø8Ø3               ;
Ø8Ø3               ; JUMP-TABLE OF ENTRY-POINTS:
Ø8Ø3               ;
Ø8Ø3 4CØ9Ø8               JMP STARTS      ;ENABLE BUFFER
Ø8Ø6 4C99Ø8               JMP CANCEL      ;DISABLE BUFFER
Ø8Ø9               ;
Ø8Ø9 ADØØEØ STARTS LDA $EØØØ       ;WHICH LANGUAGE?
Ø8ØC C94C                 CMP #$4C        ;APPLESOFT?
Ø8ØE DØ36                 BNE STEXIT      ;NO, SO QUIT!
Ø81Ø               ;
Ø81Ø A9ØA                 LDA #>BASIC     ;-> NEW START
Ø812 AØØ1                 LDY #1          ;OF BASIC (+1)
Ø814               ;
Ø814 C467                 CPY $67         ;WARM ENTRY TO ME?
Ø816 DØØ4                 BNE STCOLD      ;NO
Ø818 C568                 CMP $68         ;WARM FOR SURE?
Ø81A FØØC                 BEQ STLINK      ;YES!
Ø81C               ;
Ø81C 8467   STCOLD STY $67         ;SET UP THE NEW
Ø81E 8568                 STA $68         ;START-OF-BASIC
Ø82Ø A9ØØ                 LDA #Ø
Ø822 8DØØØA               STA BASIC       ;TRADITION
Ø825 2Ø4BD6               JSR $D64B       ;EXECUTE 'NEW'.
Ø828               ;
Ø828 A94C   STLINK LDA #$4C
Ø82A 85BA                 STA $BA
Ø82C A949                 LDA #<CHLINK    ;TIE IN TO
Ø82E 85BB                 STA $BB         ;CHRGET.
Ø83Ø A9Ø8                 LDA #>CHLINK
Ø832 85BC                 STA $BC
Ø834               ;
Ø834 A97C                 LDA #<INLINK
Ø836 8538                 STA $38         ;TIE IN TO THE
Ø838 A9Ø8                 LDA #>INLINK    ;INPUT HOOK 'KSW'
Ø83A 8539                 STA $39
Ø83C               ;
Ø83C ADEAØ3 STTIES LDA $3EA
Ø83F C94C                 CMP #$4C        ;IS DOS PRESENT?
Ø841 DØØ3                 BNE STEXIT      ;NO, NO DISK HERE!
Ø843 2ØEAØ3               JSR $3EA        ;TELL DOS ABOUT TIE-IN
Ø846 4CØ3EØ STEXIT JMP $EØØ3        ;WARM START
Ø849               ; THIS ROUTINE IS ENTERED EVERY TIME
Ø849               ; APPLESOFT FETCHES A BYTE OF BASIC.
Ø849               ;
Ø849 2CØØCØ CHLINK BIT $CØØØ       ;KEY PRESSED?
Ø84C 1Ø26                 BPL CHCOLO      ;NO, NOT YET
Ø84E 48                   PHA             ;SAVE BASIC BYTE
Ø84F 8A                   TXA             ;SAVE X-REGISTER
Ø85Ø 48                   PHA
Ø851 AEACØ8               LDX BIX         ;GET INPUT POINTER
Ø854 E8                   INX             ;PREPARE TO STEP UP
Ø855 ECADØ8               CPX BOX         ;BUT IS BUFFER FULL?
Ø858 DØØA                 BNE CHSTOW      ;NO, GO & STASH
Ø85A 98                   TYA             ;BUFFER FULL:
Ø85B 48                   PHA             ;(BELL USES Y-REG)
Ø85C 2ØE2FB               JSR $FBE2       ;RING THE BELL!
Ø85F 68                   PLA
Ø86Ø A8                   TAY
Ø861 4C6EØ8               JMP CHRETR
Ø864               ;
Ø864 8EACØ8 CHSTOW STX BIX         ;SAVE NEW POINTER
Ø867 CA                   DEX             ;-> PLACE FOR THE KEY
Ø868 ADØØCØ               LDA $CØØØ       ;GET THE KEY
Ø86B 9DØØØ9               STA BUF,X       ;SAVE IN BUFFER
Ø86E               ;
Ø86E 8D1ØCØ CHRETR STA $CØ1Ø       ;RESET KEYBOARD
Ø871 68                   PLA
Ø872 AA                   TAX             ;RECOVER X-REG
Ø873 68                   PLA             ;& BYTE OF BASIC
Ø874               ;
Ø874 C93A   CHCOLO CMP #$3A        ;(CHRGET REPLACEMENT)
Ø876 BØØ3                 BCS CHBACK      ;"
Ø878 4CBEØØ               JMP $BE         ;"
Ø87B 6Ø     CHBACK RTS             ;"
Ø87C               ;
Ø87C               ; THIS ROUTINE IS USED WHENEVER A
Ø87C               ; KEY IS NEEDED FROM THE KEYBOARD.
Ø87C               ;
Ø87C 8EABØ8 INLINK STX SAVX        ;SAVE IT
Ø87F AEADØ8               LDX BOX         ;GET OUTPUT POINTER
Ø882 ECACØ8               CPX BIX         ;ANYTHING IN BUFFER?
Ø885 DØØ6                 BNE INSEND      ;YES, GO SEND IT!
Ø887 AEABØ8               LDX SAVX        ;NO, RESTORE X-REG
Ø88A 4C1BFD               JMP $FD1B       ;NORMAL KEY HANDLER
Ø88D               ;
Ø88D 9128   INSEND STA ($28),Y     ;STOP FLASHING
Ø88F BDØØØ9               LDA BUF,X       ;GET KEY FROM BUFFER
Ø892 EEADØ8               INC BOX         ;UPDATE POINTER
Ø895 AEABØ8               LDX SAVX        ;RESTORE X-REG
Ø898 6Ø                   RTS
Ø899               ; DISABLE THE KEYBOARD BUFFER
Ø899               ;
Ø899 A9C9   CANCEL LDA #$C9
Ø89B 85BA                 STA $BA
Ø89D A93A                 LDA #$3A        ;RESTORE CHRGET
Ø89F 85BB                 STA $BB         ;ORIGINAL STUFF
Ø8A1 A9BØ                 LDA #$BØ
Ø8A3 85BC                 STA $BC
Ø8A5               ;
Ø8A5 2Ø89FE               JSR $FE89       ;EXECUTE "IN#Ø".
Ø8A8 4C3CØ8               JMP STTIES
Ø8AB               ;
Ø8AB ØØ     SAVX   .BYT Ø          ;SAVE AREA FOR X-REG
Ø8AC ØØ     BIX    .BYT Ø          ;-> PLACE FOR NEXT BYTE
Ø8AD ØØ     BOX    .BYT Ø          ;-> NEXT ONE TO DELIVER
Ø8AE               ;
Ø8AE               ; (BIX=BOX) MEANS BUF IS EMPTY
Ø8AE               ; (BIX+1=BOX) MEANS IT'S FULL!
Ø8AE               ;
Ø8AE               ; THE ABOVE MUST END BY $8FF
Ø8AE               ; OR IT WILL BE OVERWRITTEN!
Ø8AE               ;
Ø8AE               BUF=$9ØØ               ;BUFFER IS $9ØØ-$9FF
Ø8AE               BASIC=$AØØ             ;NEW START-OF-BASIC
Ø8AE               ;
Ø8AE                      .END

SYMBOL TABLE

SYMBOL VALUE

BASIC  ØAØØ   BIX    Ø8AC   BOX    Ø8AD
BUF    Ø9ØØ   CANCEL Ø899   CHBACK Ø87B
CHCOLO Ø874   CHLINK Ø849   CHRETR Ø86E
CHSTOW Ø864   INLINK Ø87C   INSEND Ø88D
SAVX   Ø8AB   STARTS Ø8Ø9   STCOLD Ø81C
STEXIT Ø846   STLINK Ø828   STTIES Ø83C
```

Program 4.

```
1ØØ REM MAKE "GEN KEYBUF"
11Ø D$ = CHR$ (4)
12Ø F$ = "GEN KEYBUF"
13Ø PRINT D$"OPEN "F$
14Ø PRINT D$"WRITE"F$
15Ø PRINT "FP"
16Ø PRINT "MON I"
17Ø PRINT "BRUN MINI-ASSM"
18Ø PRINT "8Ø3:";: REM NOTICE SEMICOLON
19Ø READ Z$
2ØØ IF Z$ = "END" GOTO 23Ø
21Ø PRINT " "Z$
22Ø GOTO 19Ø
23Ø PRINT "FP"
24Ø PRINT "BSAVE KEYBUF,A$8Ø3,L$F8"
25Ø PRINT D$"CLOSE"
26Ø END
27Ø DATA JMP8Ø9,JMP899,LDAEØØØ
28Ø DATA CMP#4C,BNE846,LDA#A
29Ø DATA LDY#1,CPY67,BNE81C
3ØØ DATA CMP68,BEQ828,STY67
31Ø DATA STA68,LDA#Ø,STAAØØ
32Ø DATA JSRD64B,LDA#4C,STABA
33Ø DATA LDA#49,STABB,LDA#8
34Ø DATA STABC,LDA#7C,STA38
35Ø DATA LDA#8,STA39,LDA3EA
36Ø DATA CMP#4C,BNE846,JSR3EA
37Ø DATA JMPEØØ3,BITCØØØ,BPL874
38Ø DATA PHA,TXA,PHA
39Ø DATA LDX8AC,INX,CPX8AD
```

```
400 DATA BNE864,TYA,PHA
410 DATA JSRFBE2,PLA,TAY
420 DATA JMP86E,STX8AC,DEX
430 DATA LDAC000,"STA900,X",STAC010
440 DATA PLA,TAX,PLA
450 DATA CMP#3A,BCS87B,JMPBE
460 DATA RTS,STX8AB,LDX8AD
470 DATA CPX8AC,BNE88D,LDX8AB
480 DATA JMPFD1B,"STA(28),Y","LDA900,X"
490 DATA INC8AD,LDX8AB,RTS
500 DATA LDA#C9,STABA,LDA#3A
510 DATA STABB,LDA#D0,STADC
520 DATA JSRFE89,JMP83C,BRK
530 DATA BRK,BRK,END
```

If you have Integer BASIC in ROM or in a Language Card, substitute:

```
150 PRINT "INT"
170 PRINT "CALL -2667" :REM MINI-
    ASSM
```

Program 5.

```
460 ML = 600 : FOR I = ML TO ML+25
470 READ X: POKE I,X: NEXT
480 LIST  (... OR PRINT SOME STUFF)
490 CALL ML
500 DATA 173,80,192,173,0,192
510 DATA 41,127,170,202,208,253
520 DATA 173,81,192,173,0,192
530 DATA 41,127,170,202,208,253
540 DATA 240,230
```
©

Micro Power Bench™

NEW!

- Single Switch Control of CPU and Peripherals
- Built in circuit breaker protects your system
- Four power expansion outlets
- Choice of high quality oak or walnut
- Compatible with IBM, APPLE, TRS-80 and others

SAVE! SAVE! ONLY $79^{00}

Order Direct
800-343-4311
Master Charge and Visa Accepted

CAB-TEK, Inc.
Riverside St., Nashua NH 03062
CIVILIZING COMPUTERS

MX 80
ACOUSTIC ENCLOSURE $99!

DEALER INQUIRIES INVITED

Atari Lister

LeRoy J. Baxter, Milwaukie, OR

Debugging a long program listing can be tedious. Most of us have typed in a long program and then had to hunt for errors when it wouldn't RUN. This utility routine can make the job a little easier.

Make a copy of this program and LIST it to tape or disk. When you need it, load it with the ENTER command (the line numbers shouldn't conflict). Then type GOTO 32700. A prompt will appear. Press RETURN, and the first set of six program lines will be LISTed to the screen, regardless of their line numbers, with spaces between the lines. Then with a touch of RETURN, it LISTs the next set of six lines.

Enter "EDIT," and the program goes to the Editing Subroutine. It asks for the line number of the offending line, then LISTs the line and prints the command "CONT" below it. You can then edit the line using the screen editor keys.

When you press RETURN, the line will be entered into the program. You can enter or delete complete lines using standard techniques. Simply move the cursor up and enter your new line between the LISTed line and CONT. When you are done, enter "ERASE," and the utility program will erase itself.

```
32700 DIM A$(5):T=0
32705 Z=0:INPUT A$:ON (A$="EDIT")+(A$
      ="ERASE")*2 GOTO 32730,32745
32710 ? CHR$(125):ADDR=PEEK(136)+PEEK
      (137)*256:FOR X=0 TO T:ADDR=ADD
      R+PEEK(ADDR+2)*(T>0):NEXT X
32715 LINENO=PEEK(ADDR)+PEEK(ADDR+1)*
      256:Z=Z+1:IF LINENO>=32700 THEN
       ? "* END OF LISTING *":GOTO 32
      710
32720 LIST LINENO:T=T+1:ADDR=ADDR+PEE
      K(ADDR+2):IF Z<6 THEN 32715
32725 GOTO 32710
32730 ? "WHAT LINE #";:INPUT X
32735 ? CHR$(125):POSITION 2,4:LIST X
      :? :? :? :? "CONT":INPUT A$:POS
      ITION 2,0:POKE 842,13:STOP
32740 POKE 842,12:T=T-6:GOTO 32705
32745 ? CHR$(125):POSITION 2,4:FOR X=
      32700 TO 32750 STEP 5:? X:NEXT
      X:? "POKE 842,12"
32750 POSITION 2,0:POKE 842,13:STOP
```
©

The arcade-warp is open!

YOU CAN RUN BUT CAN'T HIDE. YOU CAN KILL BUT CAN'T ESCAPE.

Tubeway! It's an insidious invasion route created by beings from a parallel universe—a strange, geometric universe. You're trapped on the rim as their fleet swarms out of the warp on a voyage of conquest. The battle is yours alone—and it's far from easy because normal strategy doesn't work! . . . you have to fight by their strange, geometric rules!

Here's the fastest, most fascinating of the new style space games. So involving and exciting it's destined to become an all-star, all-time hit. Be one of the first to take on the challenge of the lightfast Tubeway!

$34.95 for the Apple II*.
At your computer store or:

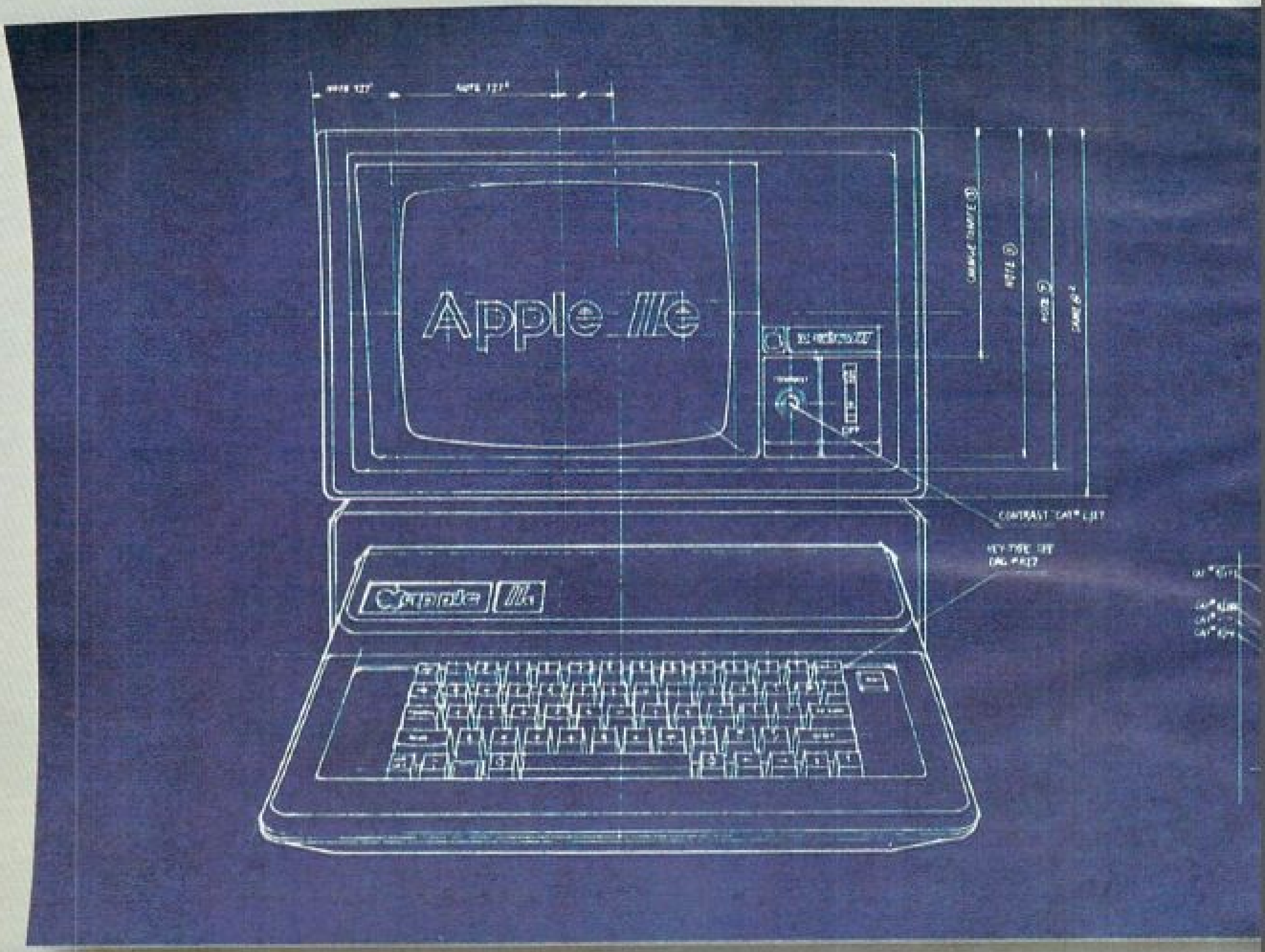
DATAMOST

9748 Cozycroft Ave., Chatsworth, CA 91311.
(213) 709-1202

VISA/MASTERCARD accepted. $2.00 shipping/handling charge. (California residents add 6½% sales tax.)

*Apple II is a trademark of Apple Computer, Inc.

It's the same old Apple II.



For years, people have been trying to build a better Apple® II.

It finally happened.

Meet the Apple IIe, an impressive new version of a most impressive machine.

The "e" means enhanced. Which means a bundle of new features:

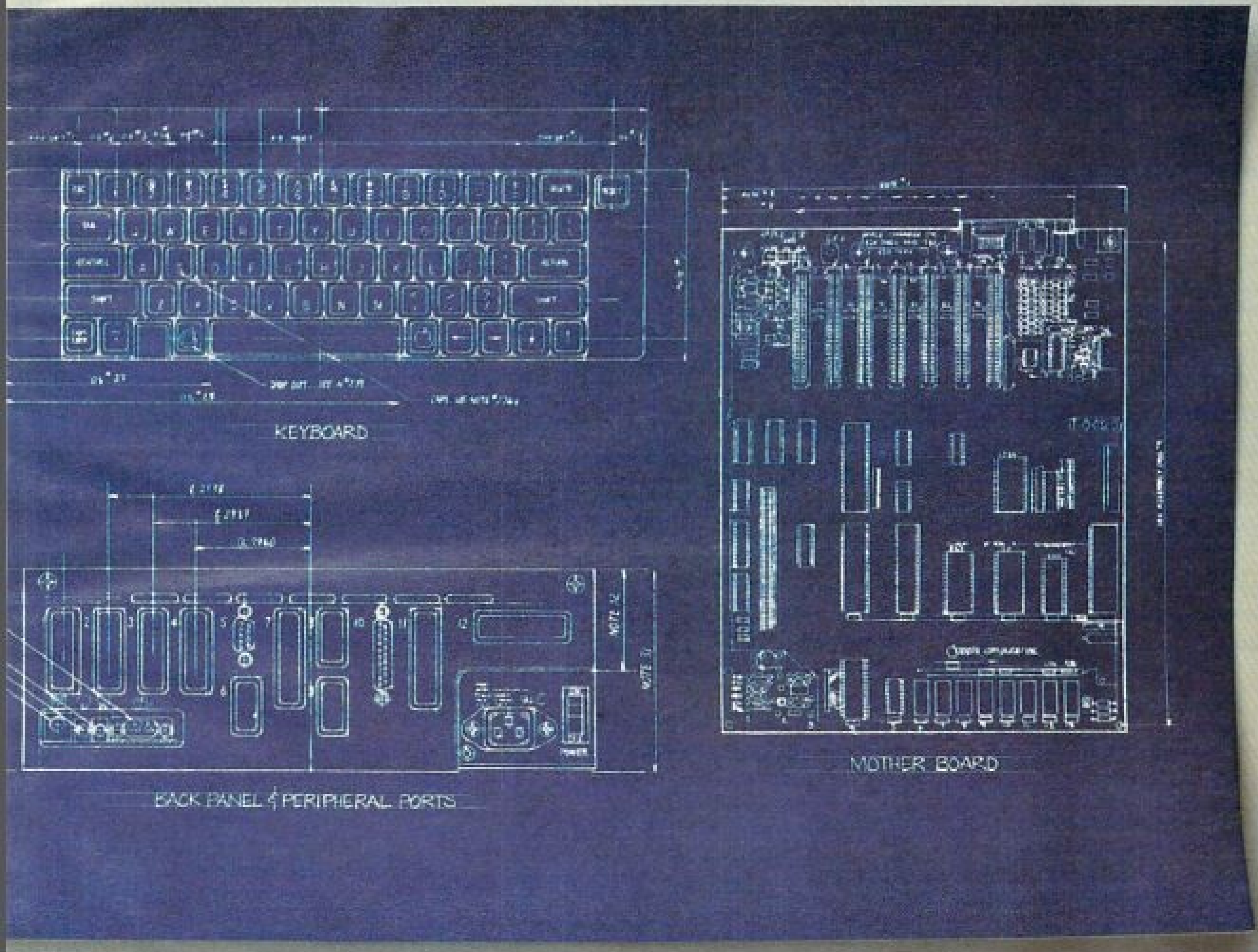
A standard memory of 64K (versus 48K) that's easily expandable. So you can create fatter files and crunch larger numbers of numbers.

A new, improved keyboard, with a complete set of ASCII standard characters. Plus full cursor controls, programmable function keys, and a rapid auto-repeat feature built into every key on the board.

Both upper and lower case characters. (And if you want to see more of them on the screen at one time, a low cost 80-column text card is available.)

Improved peripheral ports. Which make it a lot easier to connect and disconnect game controllers, printers and all those other wonderful things that go with an Apple Personal Computer.

Except for the front, back and inside.



Self-diagnostics. That's a special feature that makes it easy to give your computer a thorough check-up.

Plus an even more reliable design. Achieved by reducing the number of components — which is to say, the number of things that could go wrong.

And bear in mind, the IIe still has all those other virtues that made the Apple II so very popular. Including access to more accessories, peripheral devices and software than any other personal computer you can buy.

So visit any of our over 1300 authorized dealers, and see the newest Apple for yourself.

Like the original, it's rather extraordinary. But then some things never change.

apple

The most personal computer.

Call **(800) 538-9696** for the location of the authorized Apple dealer nearest you, or for information regarding corporate purchases through our National Account Program. In California **(800) 662-9238.** Or write Apple Computer Inc., Advertising and Promotion Dept., 20525 Mariani Ave., Cupertino, CA 95014. ©1983 Apple Computer Inc.

WORD PROCESSING HAS NEVER BEEN SIMPLER

Bank Street WRITER™

Brøderbund's Bank Street Writer turns your Apple or Atari computer into a powerful word processor, with many of the advanced features you'd expect to find only in an expensive business system. Powerful, yet purposefully simple, Bank Street Writer has no complex codes to memorize. The screen guides you every step of the way. It's everything you're ever likely to need in a word processor at a price you can afford. Here are just a few of its many features: • Add, move, insert and erase blocks of text, • Universal search and replace, • Automatic centering and indent, • Automatic word wrap, so you don't have to hyphenate or "return" at the end of each line, • Potent print format routines all in memory, • Disk storage and retrieve functions with password protection, • Document chaining allows you to print documents of unlimited length, • Page headers and automatic page numbering—top or bottom, • Highlighting of text, • Upper and lowercase without additional hardware.

Brøderbund's Bank Street Writer comes complete with Tutorial and Utility programs, a comprehensive reference manual and a free back-up disk.

Student approved, the entire system has been extensively tested by Bank Street College of Education and Intentional Educations.

Bank Street Writer. The ground-breaking, sensible combination of word processing power, thoughtful design, and exceptional value.

The First Word Processor For The Entire Family.

Hardware requirements: Apple version requires Apple II or Apple II + with 48K and Applesoft in ROM of language card, DOS 3.3. Atari 400/800 version requires 48K and BASIC cartridge. Both versions require only one disk drive.

Brøderbund Software

1938 Fourth Street, San Rafael, California 94901, Telephone (415) 456-6424

Apple Game Animation Package

Michael P. Antonovich

The *Game Animation Package* is marketed by Synergistic Software as a two-part program package: *Fast Draw* and *Micro Sketcher*. *Micro Sketcher* is for creating high-resolution color pictures which can be used as backgrounds in games or other programs. *Fast Draw* is an excellent graphics utility for creating bit-mapped shape tables which can be accessed from BASIC programs to achieve fast, smooth, flicker-free action.

Fast Draw

First, let's look at *Fast Draw*. Have you ever wanted to be able to create shape tables in color? Or to move shapes around the screen at lightning speed without screen flickering as you draw and erase the shapes? I, for one, was glad to see this package. I found *Fast Draw* a very easy way to create and manipulate shapes.

Fast Draw consists of four major program segments which allow the user to create and manipulate bit-mapped shape tables. The four segments are:

Delimit – This program is a shape table editor which allows you to create shapes using any combination of the eight standard high-resolution colors.

Examine – This program provides a very simple way of viewing each shape in the shape table as it moves across the screen at various speeds.

Placement – This allows the shapes from a table to be placed on the screen under paddle control. This option is the only way to segment a previously created shape table and to re-create it in a different form or with a different set of shapes.

Shift – This utility performs color shifts on the shapes on the screen. Shapes can also be inverted.

The Delimit program allows you to create a shape dot-by-dot in a manner somewhat similar to that used in the character generator section of the *DOS Tool Kit*. One major advantage over the *Tool Kit* is the ability to define the size of the shape in terms of the number of horizontal and vertical dots. Thus one shape is not limited to the size of a single letter. Of course, another major advantage is the ability to create colored shapes. Delimit is easy to use, and a menu is provided at the bottom of the screen in case you forget the commands or you don't read manuals. Shapes are added to the shape table one at a time.

Once a shape has been added to a table, it is relatively easy to remove that shape by using Delimit in combination with the Placement utility. Have you ever tried to combine or eliminate shapes from a regular Apple shape table by hand? With these two utilities, it is easy. Delimit also provides two modes to store shapes. There is the normal shape-saving mode for objects which require smooth movement, and a space-saving mode for objects which require smooth movement, and a space-saving mode for shapes that can move using larger jumps. In general, I found it quite easy to create fairly complicated shape tables using Delimit.

Viewing, Positioning, And Controlling Color

Examine is a simple utility for viewing the shapes in the shape table. Each shape is shown, one at a time, moving across the screen with nine different combinations of "speed" and "delay." The major problem with this utility is that if you want to see the fifth shape in the table, you must first watch the first four shapes dance across the screen nine times each. If you just want to view a shape table, it is quicker to use the Placement utility rather than Examine.

Placement allows each shape to be placed onto the screen as many times and in as many places as desired. This utility has two major uses. The first is when a background picture is needed in which a single shape is to be displayed several times in different places on the screen. Placement can place any shape anywhere on the screen as many times as necessary, after which the screen can be saved. The second major use of Placement is to edit shape tables. While a shape cannot be removed from a shape table by simply deleting the shape, the shapes you want to keep can be placed on the screen using Placement, and then reassembled into a new shape table using Delimit.

The Shift utility allows a shape's color to be shifted. Shapes can also be inverted.

Fast Draw routines are easy to access from BASIC programs. I found the instructions very clear on the methods available to access *Fast Draw* shape tables from BASIC. The only problem with the documentation is that the three demo programs listed in the manual will not work as is. However, with careful reading of the *Fast Draw* instructions, I was able to correct the demo programs. The only feature that I felt was missing from *Fast Draw* was a way to edit existing shapes. Once a shape was made, it could not be changed or used as the basis of the next shape. Therefore, if you needed a series of similar shapes, you would have to start each one from scratch.

Fast Draw (written by Glen Bredon) is an excellent graphics utility: well-written, easy to use, and well-documented. These procedures are so good that you might want to use them in your own programs. Fortunately, Synergistic Software decided not to copy-protect the diskette, so these routines can be used on any other diskette. However, Synergistic Software requests that you sign a license agreement first. There is no fee for the license agreement. That's fair enough, isn't it?

Micro-Sketcher

Micro-Sketcher is a menu-driven graphics utility for creating high-res color pictures, allowing you to create, display, edit, save, fill, and load tables to create full screens. One thing that makes *Micro-Sketcher* unique is that it allows you to create and save segments of a picture rather than having to work with the entire screen. These picture segments then can be displayed individually or in combination to create the final screen image.

I did find some problems with this package. First of all, full screen means only 256 positions horizontally, while, as we all know, the Apple screen is 280 positions in the high-resolution mode. This means there is a wide black border on the right side of the screen. This creates a problem with the fill routines, which fill out to all 280 positions. If a border is not placed around the screen, the color fill routine can cause some rather undesirable effects. In addition, once a color has been selected and an area filled, that area cannot be redefined with a new color. If a new color is desired, that sketch in the shape table will have to be redone. If you are working with the entire screen and choose the wrong color, you will have to start over or live with the color selected.

There is also no continuous draw capability. All lines are drawn as line segments by defining both end points of the line. This method is known as "rubber banding" in some packages because a flashing line is shown on the screen from the first end point to the current position. When the second end point is chosen, the line becomes solid. This is great for drawing tables, rooms, and buildings, but it is very difficult to draw curved shapes such as circles, letters, trees, etc. There are no circle utilities to create circles, or character utilities to add letters or text to your picture, either.

There is also no "paintbrush mode" such as is found in many packages which would allow you to create interesting effects such as shading, trees, bushes, and so on by using different "paint brushes."

A minor problem is that it is too easy to erase the entire screen with the "X-clear" command. After you've worked hard over a picture, a simple slip of the left hand onto the X key can make you want to bang your head against the wall. A two-key command such as CTRL-X would be far better and safer.

The edit mode of *Micro-Sketcher* is unusual. To edit a shape, the program removes one line at a time from the end of the shape. Therefore, if an error was made at the beginning of the shape, all of the lines must be removed until you get back to the line in error, and then the lines must be redrawn. Also, the edit mode may not remove all of the dots from the screen as it removes the lines. These remaining dots cannot be edited out of the picture with this package. You cannot simply draw over these dots with a black pen, because you can only draw white lines on a black background. Start over with a clean screen.

On the positive side, *Micro-Sketcher* has a fast and very efficient fill routine (written by John Conley) which is capable of handling fairly complex shapes. In fact, the fill routine is much better than those in many other graphics packages.

Except for the X key, the program has good protection against faulty input. The documentation is good, but not as clear as the *Fast Draw* documentation. Up to 32 colors are available for the fill routines, and the author has split these colors into compatible groups to eliminate the problem of color smearing when two colors are placed next to each other.

Another nice feature is the use of game paddles or a joystick to roughly position a point, and the use of the I, J, K, or M keys to disable the paddles or joystick and make fine adjustments.

In general, the *Game Animation Package* is well worth the price for people who would like to write animated games, but who do not know 6502 machine language. The *Fast Draw* routines are worth the price of the package themselves for that purpose. While the documentation is fairly good, it does help to first have an understanding of the way the Apple uses graphics and the graphics screens. However, the shape tables created by these two packages are not the same type of shape tables described in the *Apple Reference Manual* or in some other Apple books.

You must use the routines provided on the *G.A.P.* diskette to be able to draw these shapes. In fact, the shape tables created by the two different methods are not really compatible with each other (or at least I was not able to use them interchangeably). However, since the manual explains how to access these shape tables from BASIC, and since the routines are on the diskette, let's go out and add some animation to our games.

Game Animation Package
Synergistic Software
5221 120th Avenue SE
Bellevue, WA 98006
$49.95

©

Moptown – Educational Games For Apple

Sheila Cory

Colorful blocks of varying sizes and shapes that are found in many elementary school classes are called Attribute Blocks. They are used to stimulate rational thinking by giving children experience in distinguishing attributes and carrying out logical operations.

The "Moppets" who live in *Moptown* are computerized Attribute Blocks, with each of the inhabitants identifiable by their peculiar combination of traits.

The traits used to identify the 16 Moppets who populate *Moptown* are: (1) tall or short, (2) fat or thin, (3) red or blue, and (4) Bibbit or Gribbit. All Bibbits have big noses and big feet, and all Gribbits have tails. Like work with Attribute blocks, games in *Moptown* involve logical thinking. Working with attributes on the computer allows, among other things, the random assignment of the attributes, feedback as to the correctness of response, and immediate reinforcement.

Moptown

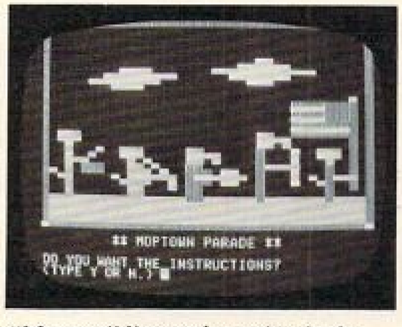
Moptown is a program designed for elementary school-aged youngsters. Programmed by Leslie Grimm (whose excellent programs – *Bumble Plot*, *Bumble Games*, and *Juggles' Rainbow* – were reviewed in **COMPUTE!** recently), this set of programs consists of 11 different games that develop the ability to identify and isolate attributes. The games are carefully sequenced from easy to hard, providing an ideal structure for understanding and learning. The programs would be appropriate for use in kindergarten through grade six, with the most difficult even providing challenging fun and learning for children in junior high school.

Recognition Games: Easy To Difficult

In Make My Twin, the simplest of the games, the user looks at a *Moptown* villager (or Moppet, as they're called), and then describes its four attributes in order to make its twin. To do this, the child needs to be able to separate each of the attributes from the whole – an excellent activity for the development of analytical thinking. To save typing, the program allows the child to use a one-letter input to describe the attribute. With young children, this can be a very important feature in a program, yet one that some programmers forget to consider.

Who's Different? lines four Moppets up assembly-line style and asks the user to find the one that is different. After identifying the different Moppet, the user then must identify which of the four attributes makes it different. Another possibility in this game is to have four different Moppets drawn, and have the user choose which one is *most* different. This variation is considerably more difficult than the previous one.

What's the Same? is similar to the previous game, except the object is to find the one attribute the Moppets have in common. As in the other games, no help is given if the user continually selects the wrong answer. This could be a problem for a child who chooses to play a game that is beyond his or her level of skill.



```
** MOPTOWN PARADE **

DO YOU WANT THE INSTRUCTIONS?
(TYPE Y OR N.)
```

"Moppets" line up for review in the Moptown Parade game.

Who Comes Next? is a pattern recognition game. There are three possible patterns: ABABAB, ABBABB, or AABAAB. Four Moppets are lined up; the user determines the pattern and then describes what the fifth Moppet should look like. The task involves not only identifying the pattern, but also dissecting the appropriate Moppet into its four attributes in order to describe them. If the Moppet is described incorrectly, it is drawn the way it was described and the user again has an opportunity to describe the Moppet correctly.

User-Determined Patterns

The next game is Moptown Parade. Like all of these games, it is introduced with an appropriate picture and song – in this case, "She's a Grand Old Flag!" The object of this game is to create the participants in a parade according to a rule determined by the user.

The rule establishes how many traits each successive Moppet in the parade should have that are different from those of the previous Moppet. For example, if the rule is "1", then the next Moppet in the parade will differ from the Moppet in front of him by just one trait. If Moppet 1 is tall, blue, fat, and a Gribbit, then Moppet 2 could be tall, red, fat, and a Gribbit. If a mistake is made, the incorrect Moppet is drawn and then erased so the user can try again.

Who's Next Door? makes trait analysis of two Moppets an essential step for determining the second Moppet of another set. The first pair of Moppets are compared to see which single trait is different. A third Moppet is shown, and its pair must be described so that the two differ in the same attribute as the first pair.

In My Secret Pal, the user selects four traits to describe a

Moppet. The program responds by drawing the Moppet described, and then telling how many of those traits are correct to describe the secret pal. This game is quite a challenge, as the program does not tell you *which* traits are correct, only *how many* are correct. It is up to the user to develop good guessing strategies!

Careful trait analysis is necessary to be successful in the next game, Change Me!. In this game, four boxes are drawn on the screen. A Moppet is drawn in box one and box four. Again, as in Moptown Parade, a rule of "1" or "2" determines how many trait differences there should be in each successive Moppet. The problem is to determine what the second and third Moppet should look like in order for the fourth Moppet to have just the specified number of different attributes.

Clubhouse is more difficult still, requiring logical deductions to decide which Moppet can join the Moppets Club. Each time a Moppet is described, the program responds by telling whether or not he can join the club. The object of the game is to figure out what rule or rules are being applied to each Moppet to either accept him into or reject him from the club.

The last two programs, Moptown Map and Moptown Hotel, carry the skills developed in the previous games a step further. In both of these games, the user has to be concerned with attributes shared by Moppets in the same row *and* the same column. Thinking of relationships in two dimensions makes these two games substantially more difficult than the previous ones; but with mastery of the earlier games, these should be challenging enough to be interesting, yet easy enough to be fun.

Color Monitor Crucial

The documentation for *Moptown* is clear and concise. I disagree with the claim that these programs are suitable for use with a black and white monitor, however. Color is crucial to these programs, as it is one of the four attributes by which the Moppets are distinguished from each other. As the manual states, it *is* possible to discern the differences on a black and white monitor, but I feel it makes the games too difficult. One outstanding feature of the manual is the inclusion of suggestions on how to use these programs when there is just one computer for a whole class.

Sound adds a lot to this program. However, sound can be a distraction in some classroom situations. The program does not have a "sound/no sound" option, which might make it inappropriate for some classes. The program also makes different sounds when a child gets an answer correct than when he or she gets an answer wrong. Some children could be very upset about having others know how they're doing when they're working so hard to master a difficult concept.

How Children Rate *Moptown*

Because it is difficult for me to assess how kids would respond to a program, I gathered a group of "kid consultants" to test out these programs. Bret, 11 years old, spent about two hours on *Moptown*. He said he enjoyed all the games, but felt his friends would most enjoy Moptown Hotel, which is the most difficult. He said he would like to borrow the programs from me in order to have more time with them.

Cara, ten years old, enjoyed all of the programs *except* Moptown Map and Moptown Hotel, which she felt were too difficult. She had only a little more than an hour to spend on the programs, so she would possibly enjoy those difficult ones more if she could work with the games a bit longer. Cara felt her friends would enjoy Clubhouse the most. Like Bret, she asked if she could borrow the diskette for more work with these programs.

Chrissa, eight years old, loved the games. She thought Make My Twin was a little boring because it was too easy, but enthusiastically endorsed Clubhouse. The Kids all tended to ask adults how to play the games rather than read the instructions. In a classroom situation, it would be a good idea for the teacher to introduce each of the games to the whole class before having the children play individually.

Moptown runs on an Apple II Plus with 48K. It comes on diskette, with back-up diskette and manual included in a handy package.

Moptown
Apple Computer, Inc.
20525 Mariani Avenue
Cupertino, CA 95014
$50

COMPU SENSE

CARDRITE

A graphic light pen for the VIC-20®.

Supplied with a cassette of six programs.

$29.95

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

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

® VIC-20 is a registered trademark of Commodore, Inc.

MasterCard VISA

Easy Apple Editing

Roland Brown

This editor routine provides a powerful utility for Applesoft programmers: the ability to easily modify BASIC program lines.

The Apple II+ was created for its advanced BASIC. Programming in Applesoft is much better and easier than with integer BASIC. One main problem, however, with Applesoft is its editing. Some people invest in a ROM editor, others create their programs using a text editor, and others just suffer with the frustrating ESCape codes. Presented here is a 3/4K machine language program for the Apple II+ 48K or equivalent with DOS 3.3.

The BASIC Line Editor will not destroy the current BASIC program, but will destroy its strings. Once saved on disk as a binary file, Editor can be loaded into memory by the command:

```
]BRUN B.L.E.,A$9A00
```

To edit a BASIC program line, type

```
]& (line number)
```

for example,

```
] &100
```

This will clear the screen, display the line, and place the cursor at the top left of the screen. The line is displayed in a different format from Applesoft's. The differences are: the line is continuous instead of centered on the screen, there are no spaces in the line except between quotes, and all control characters are displayed in inverse.

In the Editor numerous commands are available to you. These commands edit the line:

CTRL-B	block back
-C	convert hex to decimal
-D	delete
-F	block forward
-H	back arrow
-I	insert
-M	return
-S	search
-T	truncate
-U	forward arrow
-V	verbatim
ESC	return to BASIC

CTRL-B moves the cursor back to the previous colon, or if there is no previous colon, the beginning of the line.

CTRL-C clears the bottom of the screen, places a $ prompt on the screen and allows a line to be input. This line is converted to decimal, printed, and the cursor is returned to its original position on the line. This can be used to convert bytes in hex that need to be POKEd/PEEKed.

CTRL-D deletes the character at the cursor.

CTRL-F moves the cursor to the next colon, or if there is no next colon, to the end of the line.

CTRL-H (back arrow) moves the cursor back one space.

CTRL-I inserts a space in the line at the cursor position. Note: CTRL-I will not insert at the top left of the screen.

CTRL-M (return) can be entered at any place in the line and the entire line will be entered into the program.

CTRL-S searches for the next character entered.

CTRL-T truncates the line at the cursor position, so the cursor is at the end of the line + 1.

CTRL-U (forward arrow) moves the cursor forward one space.

CTRL-V allows raw control characters to be entered into the line. This can be used to enter returns or backspaces for easier printing control.

ESC will exit the Editor with the line untouched (so if you make a mistake, the line is not lost).

The Editor provides a < at the left side of line 7 as a guide to where BASIC truncates its lines. This will not affect your line if you pass the line through it.

Typing It In

Program 1 is a BASIC program which loads the machine language for the Editor into memory. If the DATA statements have all been entered correctly, the program will provide instructions for saving the binary file to the disk, which should be done immediately. If an error is detected in the data, the program will stop. If this happens, check your DATA statements carefully, correct all mistakes, and run the program again.

Program Explanation

Loading the program resets the stack to the same level as BASIC does, sets up the ampersand vector ($3F5), clears the screen, moves the DOS buffers down so it is safe, prints the title and restarts BASIC.

The entry of the program uses BASIC routines to read in the line and find the line in memory. If the line is not there, the program returns to

BASIC.

The line is disassembled into the input buffer ($200-$2FF) by using a modified version of CHRGET. If the character is text, CHRGET places it in the line. If the character is one representing a command, CHRGET looks it up in the table of BASIC commands and puts the command name in the buffer. Once the end of the line is reached, CHRGET enters the edit section of the program.

The edit program displays the line, gets a character from the keyboard, and processes it. Explanations of the different commands would take too long, but fairly adequate documentation of the program's workings can probably be understood by many Apple owners. If you do not understand any of the Editor commands, just experiment with them for a while.

```
100  FOR I = 39424 TO 40065: READ A:CK = CK +
     A: POKE I,A: NEXT
110  IF CK <  > 88754 THEN  PRINT "ERROR IN
     DATA STATEMENTS": STOP
120  HOME : PRINT : PRINT "BASIC LINE EDITOR
      INSTALLED AT": PRINT "LOCATIONS 39424-
     40065 ($9A00-9C81)": PRINT
130  PRINT "TYPE 'BSAVE B.L.E.,A$9A00,L$282'
     ": PRINT "TO STORE BINARY FILE": PRINT
140  PRINT "TYPE 'CALL 39424' TO ACTIVATE"
150  NEW
200  DATA  162,251,154,169,76,141,245,3
210  DATA  169,43,141,246,3,169,154,141
220  DATA  247,3,32,88,252,169,153,141
230  DATA  1,157,32,212,167,169,12,133
240  DATA  36,169,112,160,156,32,58,219
250  DATA  76,208,3,32,12,218,32,26
260  DATA  214,176,1,96,165,155,133,184
270  DATA  165,156,133,185,32,88,252,32
280  DATA  224,158,104,104,32,95,156,32
290  DATA  95,156,170,32,95,156,32,36
300  DATA  237,162,5,189,0,4,41,127
310  DATA  157,0,2,202,16,245,162,6
320  DATA  134,36,32,95,156,240,61,48
330  DATA  7,157,0,2,232,76,98,154
340  DATA  160,208,132,254,132,255,160,0
350  DATA  41,127,133,253,240,15,177,254
360  DATA  8,200,208,2,230,255,40,16
370  DATA  245,198,253,208,241,177,254,8
380  DATA  200,208,2,230,255,157,0,2
390  DATA  232,40,16,241,41,127,157,255
400  DATA  1,76,98,154,134,252,170,133
410  DATA  250,133,251,32,88,252,160,0
420  DATA  185,0,2,201,32,144,2,9
430  DATA  128,32,240,253,200,196,252,144
440  DATA  239,169,188,141,0,7,165,250
450  DATA  133,36,165,251,32,90,156,32
460  DATA  12,253,201,136,208,22,202,198
470  DATA  250,16,230,160,39,132,250,198
480  DATA  251,16,222,232,160,0,132,250
490  DATA  230,251,240,213,201,149,208,30
500  DATA  164,250,200,132,250,232,228,252
510  DATA  144,7,134,252,169,32,157,0
520  DATA  2,192,40,144,188,160,0,132
530  DATA  250,230,251,76,193,154,201,141
540  DATA  208,20,166,252,169,0,157,0
550  DATA  2,32,81,168,32,88,156,162
560  DATA  255,160,1,76,68,212,201,137
570  DATA  208,39,224,0,240,32,134,249
580  DATA  166,252,230,252,240,20,189,0
590  DATA  2,157,1,2,202,228,249,176
600  DATA  245,232,169,32,157,0,2,76
610  DATA  171,154,166,249,198,252,76,193
620  DATA  154,201,132,208,24,228,252,176
630  DATA  245,198,252,134,249,189,1,2
640  DATA  157,0,2,232,228,252,144,245
650  DATA  166,249,76,171,154,201,130,208
660  DATA  38,224,0,240,18,198,250,16
670  DATA  17,160,39,132,250,198,251,16
680  DATA  9,160,0,132,251,132,250,76
690  DATA  193,154,202,240,250,189,0,2
700  DATA  201,58,208,225,76,193,154,201
710  DATA  134,208,12,228,252,176,232,169
720  DATA  58,32,58,156,76,193,154,201
730  DATA  147,208,11,32,12,253,41,127
740  DATA  32,58,156,76,193,154,201,150
750  DATA  208,18,32,12,253,201,160,176
760  DATA  109,41,127,157,0,2,32,240
770  DATA  253,76,240,154,201,148,208,5
780  DATA  134,252,76,171,154,201,131,208
790  DATA  66,134,249,162,0,189,0,2
800  DATA  157,0,187,232,208,247,169,8
810  DATA  133,37,160,0,32,66,252,169
820  DATA  164,133,51,32,103,253,32,199
830  DATA  255,32,167,255,32,142,253,169
840  DATA  189,32,240,253,165,63,166,62
850  DATA  32,36,237,162,0,189,0,187
860  DATA  157,0,2,232,208,247,166,249
870  DATA  76,193,154,201,155,240,21,201
880  DATA  160,144,11,32,240,253,41,127
890  DATA  157,0,2,76,240,154,32,226
900  DATA  251,76,193,154,32,88,156,76
910  DATA  208,3,133,255,164,250,200,132
920  DATA  250,192,40,144,6,160,0,132
930  DATA  250,230,251,232,228,252,176,7
940  DATA  189,0,2,197,255,208,229,96
950  DATA  169,15,133,37,76,34,252,230
960  DATA  184,208,2,230,185,160,0,177
970  DATA  184,56,233,48,56,233,208,96
980  DATA  194,193,211,201,195,160,204,201
990  DATA  206,197,160,197,196,201,212,207
1000  DATA  210,0
```
©

ATTENTION PROGRAMMERS!!

DATASOFT is currently seeking programs and programmers to add to their rapidly growing and expanding operation. A leading marketer and developer of personal computer software, DATASOFT offers experienced assembly-language programmers the opportunity to join their staff to develop and translate arcade games such as ZAXXON™, as well as to author original material for their games, education and home management product lines. DATASOFT pays competitive salaries, plus bonuses based on product performance. Relocation assistance is available, if needed.

If you have working knowledge of Atari, Apple, TI, or Commodore operating systems, graphics, animation and sound, call or write Melinda Storch at:

Datasoft Inc.®
COMPUTER SOFTWARE

9421 Winnetka Ave.
Chatsworth, CA 91311
(213) 701-5161 / (800) 423-5916

ZAXXON and SEGA are registered trademarks of Sega Enterprises.
DATASOFT is a registered trademark of Datasoft, Inc.

Comets

Chris Williams

For Applesoft on a 48K Apple II, this simulation of a comet's motion in high-res allows you to alter several variables. You can even send the comet into deep space.

In this article we'll be concentrating on comets. Comets have a couple of characteristics that make them well-suited to illustrate several concepts embedded in the program.

The first of these is their long periods; you have plenty of time to see what's going on. Comets can take hundreds of years to complete one orbit. Fortunately, we won't have to wait that long.

Second, they have highly elliptical orbits. Large variances in the comet's distance from the star are a visual plus.

The program makes use of both these traits to demonstrate idiosyncrasies of cometary motion. It is written in Applesoft to run on an Apple II+ (48K). The Apple, of course, uses a BASIC interpreter. If areas of code (especially in the main execution loop) look strangely written, it's because the program was designed for speed. REM statements are also placed with speed in mind.

Newton's Laws

The program, unlike most celestial simulations, does not directly use Kepler's laws. Instead, Newton's gravity equations are applied in two dimensions to drive the movements of a high-res dot which represents the comet.

A delta time interval (DT) of 120 days is used to get things done in a reasonable amount of time. Having a 120 day DT has some interesting ramifications. But we'll touch on that later.

Operation of the program is straightforward. It opens with a brief introduction, and then gives some suggestions for input parameters that will produce a stable, visually pleasing orbit. After the last input parameter is entered, execution begins.

The screen goes into high-res, a sprinkling of stars appears to set the mood, and the update loop starts. A clicking sound with distance-dependent pitch is also produced, again merely for effect.

The values used for constants and variables are not arbitrary. All numbers in the program – those the user inputs and those displayed at the bottom of the screen – have meaning. The mass of the central bright star (cross) is equal to the sun's in all calculations. The comet's mass is a plausible 1000 kgs. The screen scaling is such that its edge represents a radius just outside Pluto's orbit.

One last point of interest. If you input the following parameters:

DX = 5555
DY = 0
VX = 0
VY = 1

you'll see some strange behavior. The comet will curve inbound, pass very close to the star, and then whip right off the screen.

You Can Lose The Comet

This can be traced back to the 120 day DT value mentioned previously. As the comet gets in close to the star, its velocity increases tremendously. As a result, there are passes through the execution loop in which a very large velocity is applied over 120 days. This yields a relatively large distance traveled from the star at the completion of that pass.

Gravity is an inverse-square relationship. With a large distance *and* a high velocity, there is not sufficient attractive force to keep the comet in orbit.

This doesn't happen in nature. It is simply a peculiar effect of large DTs in numerical integration. There are many cures, but I chose to leave it alone, as a demonstration.

Try it out and experiment. I've found some unusual input combinations that seem to be on the threshold of the above problem. They result in a semi-spiral until the comet gets too close to the star and streaks out of the system.

This doesn't happen in nature either. It's just another illustration of the need for care when creating an accurate simulation.

```
10  REM  *** COMETS ***
11  REM  BY CHRIS WILLIAMS
13  REM  **************
20  ONERR  GOTO 370
30  HOME
35  REM  GO TO INTRODUCTION SUBRO
     UTINE
40  GOSUB 350
45  REM  ENTER INPUT PARAMETERS
47  REM  AND SET UP THEIR UNITS
```

```
50  INPUT "ENTER DX(X 10^6 KM)";D
     X: PRINT
60 DX = DX * 10 ^ 9
70  INPUT "ENTER DY(X 10^6 KM)";D
     Y: PRINT
80 DY = DY * 10 ^ 9
90  INPUT "ENTER VX(KPS)";VX: PRINT
100 VX = VX * 10 ^ 3
110  INPUT "ENTER VY(KPS)";VY: PRINT
120 VY = VY * 10 ^ 3: HGR
125  REM  PLOT THE CENTRAL STAR A
     S A +
130  HCOLOR= 3: HPLOT 140,80: HPLOT
     141,80: HPLOT 140,81: HPLOT
     139,80: HPLOT 140,79
135  REM  NOW SPRINKLE STARS FOR
     MOOD
140  FOR RD = 1 TO 100:X =  RND (
     1) * 279:Y =  RND (1) * 159:
      HPLOT X,Y: NEXT
145  REM  SET GRAV. EQN. CONSTANTS
146  REM  AND DT=120 DAYS, ALSO
147  REM  HI RES SCALING
150 MS = 329390 * 5.98 * 10 ^ 24:
     G = 6.67 * 10 ^ ( - 11):DT =
     120 * 3600 * 24:SXCALE = 279
      / (2 * (5900 * 10 ^ 9)):SYC
     ALE = 159 / (2 * (5900 * 10 ^
     9))
155  REM  PLACE VARIABLE LABELS
156  REM  AT BOTTOM OF PAGE
160  VTAB 22: HTAB 25: PRINT "VX=
     ": VTAB 23: HTAB 25: PRINT "
     VY="
170  VTAB 22:: PRINT "DX=": VTAB
     23:: PRINT "DY=  "
175  REM  CM IS COMET MASS IN KGS.
177  REM  CR IS SCREEN SIZE IN ME
     TERS
179  REM  OTHER CONSTANTS FOR SPE
     ED
180 CM = 1000:CR = 5900 * 10 ^ 9:
     ZERO = 0:THREE = 3:T2 = 22:T
     3 = 23:FR = 4:T8 = 28:RE = 1
     .49 * 10 ^ 11:TLL = 9 * 10 ^
     11
185  REM  LOOP STARTS AT 190
186  REM  NO COMMENTS WITHIN
187  REM  FOR SPEED
190 SS = (DX * DX) + (DY * DY):SQ
      =  SQR (SS)
200 F = CM * MS * G / SS
210 AX =  - F * (DX / SQ) / CM
220 AY =  - F * (DY / SQ) / CM
230 VX = VX + (AX * DT)
240 VY = VY + (AY * DT)
250 DX = DX + (VX * DT)
260 DY = DY + (VY * DT)
270  VTAB T2: HTAB FR: PRINT DX;"
      ": VTAB T2: HTAB T8: PRINT
     VX: VTAB T3: HTAB FR: PRINT
     DY;" "
280  VTAB T3: HTAB T8: PRINT VY
290  HCOLOR= ZERO: HPLOT XNU,YNU
300 XNU = (DX + CR) * SXCALE
310 YNU = (DY + CR) * SYCALE
320  HCOLOR= TH: HPLOT XNU,YNU
325  GOSUB 700
330  GOTO 190
340  STOP
342  REM  GOSUB 700 AT 325
343  REM  IS "CLICK" ROUTINE
350  HTAB 17: PRINT "COMETS": PRINT
     : PRINT "   THIS PROGRAM IS
     A SIMULATION OF THE": PRINT
     : PRINT "ORBITAL TRAJECTORIES
      CHARACTERISTIC OF": PRINT :
      PRINT "COMETS."
352  PRINT : PRINT "SUGGESTED INP
     UTS:DX=5555,DY=0,VX=0,VY=3":
      PRINT : PRINT
355  REM  360 CONTAINS ASSMBLY
356  REM  LOAD OF CLICK ROUTINE
360  PRINT : PRINT "HIT ANY KEY W
     HEN READY": GET A$: HOME : PRINT
     : PRINT : FOR DP = 771 TO 78
     9: READ DA: POKE DP,DA: NEXT
     : RETURN
365  DATA  173,48,192,136,208,4,1
     98,1,240,8,202,208,246,166,0
     ,76,3,3,96
368  REM  370 IS WHERE YOU GO WHEN
369  REM  ERROR FROM OFF SCREEN
370  HOME : TEXT : FOR YY = 1 TO
     10: PRINT  CHR$ (7): NEXT YY
     : HOME :: PRINT : PRINT "OKA
     Y, PAL.  ONE OF THREE THINGS
      JUST": PRINT : PRINT "HAPPE
     NED.": PRINT
380  PRINT "EITHER YOUR INITIAL V
     ELOCITIES": PRINT : PRINT "W
     ERE TOO LARGE OR YOU PASSED
     TOO": PRINT : PRINT "CLOSE T
     O THE STAR.  PASSING TO CLOS
     E": PRINT : PRINT "TO THE ST
     AR CAUSES PROBLEMS WITH A": PRINT
     : PRINT "120 DAY LOOP INTERV
     AL."
390  PRINT : PRINT "OR PERHAPS YO
     U JUST MESSED UP.": PRINT : PRINT
     "IN ANY CASE, TRY AGAIN.": END
700  POKE 1,3: POKE ZE,(T8 * SQ /
     CR) + FR: CALL 771: RETURN
```
©

Use the handy reader service cards in the back of the magazine for information on products advertised in **COMPUTE!**

A FORTH/BASIC Benchmark Test

Michael F. Heidt

This article has a twofold purpose. First, it makes a timing comparison between FORTH and BASIC by comparing runtimes for a benchmark program. Second, it demonstrates FORTH's extensibility by the implementation of a simple integer array.

Benchmarks are frequently used in acceptance testing mainframe computers. The BASIC Benchmarks used by Rugg and Feldman (*Kilobaud*, June 1977) became so popular that they were frequently used in advertising implementations of BASIC. Benchmark 7 from the *Kilobaud* article (Program 1) is the most comprehensive and was chosen for this comparison.

A quick look will show you that the program doesn't actually do much. The variable K is used as a loop counter. M is a simple array into which the values calculated in line 510 are to be stored. The subroutine at line 820 doesn't do anything. The object here is to measure the overhead required by calling a subroutine. The print statements at lines 300 and 700 allow you to start and stop a stopwatch to time the benchmark. Program 2 is the FORTH equivalent of Program 1 (the BASIC program).

The Results

BASIC	FORTH
27.43	13.58

Benchmark 7 results (seconds)

The above figures show the speed comparisons for the two versions of Benchmark 7. The measurements were made on an OSI C4-P running a 6502 processor at two megahertz. Each benchmark was run ten times and the results then averaged. This was done to average out variations in reaction time in starting and stopping the stopwatch.

As you can see from the table, the FORTH version is twice as fast as the BASIC version. The FORTH version could be made even faster by leaving out error checking, an option not available in BASIC.

It should be noticed that the FORTH version does not have a GOTO statement. FORTH has no GOTO. The structure of the FORTH program is "bottom up." This means that the most primitive sections are built first, then the next level uses the primitives and so on until the desired functions are built. However, it is possible to do "top down" programming in FORTH.

In fact, this is really how it should be done. For example, I essentially wrote the word B7 first, then added the more primitive routines. By doing it this way, you know what primitives to write, what variables will be needed, and you get some idea of just how big the job is going to be.

If you're not familiar with FORTH, the program presented here may appear complicated compared to the BASIC version. However, you should keep in mind that in addition to creating the benchmark, I have extended FORTH here to include a general integer array capability that can be used by other programs.

Program 1.

```
10 REM BENCHMARK 7,
   Kilobaud #6 p66
300 PRINT"START"
400 K=0
430 DIM M(5)
500 K=K+1
510 A=K/2*3+4-5
520 GOSUB820
530 FOR L=1 TO 5
535 M(L)=A
540 NEXT L
600 IF K<1000 THEN
    500
700 PRINT"END"
800 END
820 RETURN
```

Program 2.

```
SCR # 96
( BENCHMARK 7  WITH INTEGER ARRAYS                  MFH 1/11/81)
FORTH DEFINITIONS DECIMAL
: DIM <BUILDS DUP , 2 * 2 + ALLOT DOES> ;
: RANGE DUP ROT DUP ROT @ > ;
: READ RANGE IF ." RANGE ERROR " CR DROP @ (LEAVES MAX DIM
  IF ERROR )     ELSE 2 * + 2 + @ ENDIF ;
( READ WANTS ELEMENT NAME, E.G. E M READ - LEAVES CONTENTS )
: ADD RANGE IF ." RANGE ERROR" CR DROP DROP ELSE 2 * 2
 + + ! ENDIF ; ( WANTS VALUE ELEMENT NAME )
  0 VARIABLE K 0 VARIABLE A
: START ." START " ; : STOP ." END " CR CR ; : GOSUB ;
5 DIM M ( CREATE ARRAY M WITH 5 ELEMENTS )
: K+ K @ 1 + DUP K ! ; ( INCREMENT VARIABLE K BY ONE )
: B7 START 0 K ! BEGIN K+ DUP 2 / 3 * 4 + 5 - A ! GOSUB
        6 1 DO A @ I M ADD LOOP 1000 = UNTIL STOP ;
;S
```

©

FIRST THINGS FIRST. LEARN ALL ABOUT IT!

When you don't know the first thing about your new Apple II* you need a friendly, cheerful, easy going teacher at your side. And the ELEMENTARY APPLE is just that kind of book.

It sweeps away the confusion—explains your Apple in everyday language—shows you how to hook it up, how to use the keyboard and work on the screen.

Gently and carefully it gives you an understanding of all the things your Apple can do. And then, it even shows how easy it is for anyone to write a simple program—provides common sense answers about graphics, utility programs, and the how and why of word processors, business programs and hardware like printers.

Yes, there's a lot of information. But, not one chapter or one word is dull or difficult to follow or complicated. Prove it to yourself. Visit your computer store. Open the ELEMENTARY APPLE. Read a page of the introduction, then flip it open anywhere and read a paragraph or so. You'll find it's as understandable, as helpful and as marvelous as we say.

If you, or a member of your family, is an Apple beginner, this is the book you need. It'll teach you everything you want to know, in the way you want to learn.

Only $14.95. At computer and book stores, or:

DATAMOST INC

9748 Cozycroft Ave., Chatsworth, CA 91311. (213) 709-1202.

VISA/MASTERCARD accepted. $2.00 shipping/handling charge.
(California residents add 6½% sales tax)
*Apple II is a trademark of Apple Computers, Inc.

Not all business
And we've got the

As you know, one picture is worth a few thousand numbers.

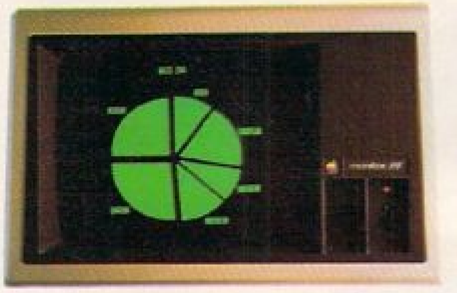
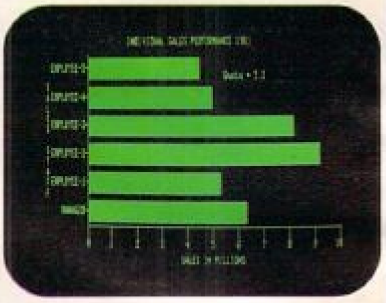
As you may not know, Apple® Business Graphics software can generate more types of pictures, in more colors, using more data than any other graphics package.

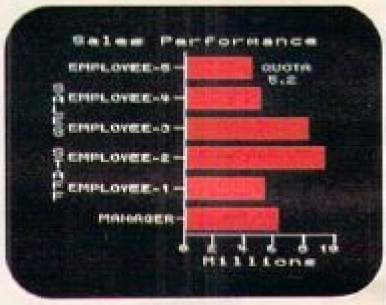
So you not only get the usual bar graphs and pie charts. You also get unusual bar graphs and pie charts. Sophisticated line and area graphs. Even scattergrams. All teamed with extremely useful and powerful features—exploded views, unlimited overlays, floating titles and more.

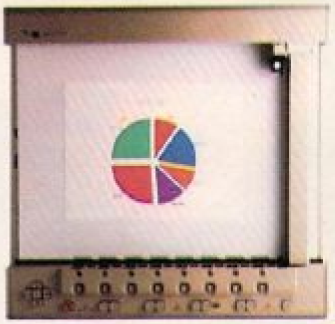
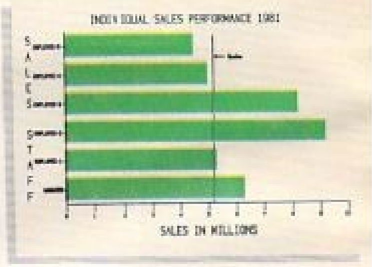
	Apple	VisiTrend/ VisiPlot	pfsGraph
Graph Types			
Line	Yes	Yes	Yes
Vertical Bar	Yes	Yes	Yes
Horizontal Bar	Yes	No	No
Side-by-side Bar	Up to 4	2	4
Pie	Yes	Yes	Yes
Partial Pie	Yes	No	No
Scattergram	Yes	Yes	No
Curve Fitting	5 Kinds	1	None
Data Points (Max.)	3500+	645	36
Plotter Compatible	Virtually Any	None	H-P7470A Only
Compatible File Types	Pascal BASIC VisiCalc	BASIC VisiCalc	pfs VisiCalc
Math Functions	Yes	Yes	No
Available Colors	6	4	4

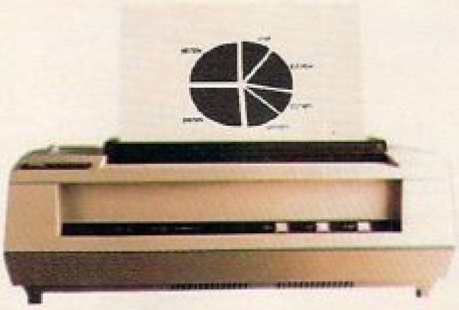
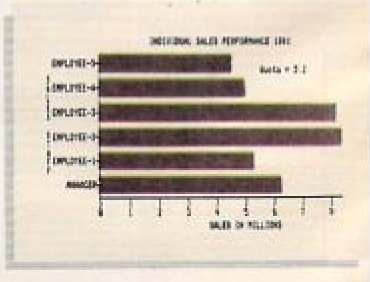
Apple Business Graphics is available for both the Apple II and Apple III.

Equally important, with our graphics package you'll find more ways to see what you're doing. On the monitor of your choice. And on virtually any printer or plotter on the market.

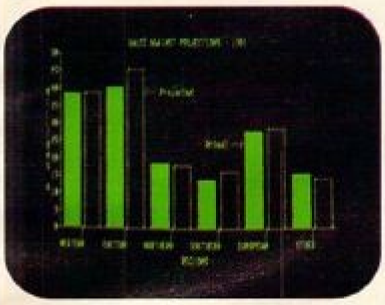



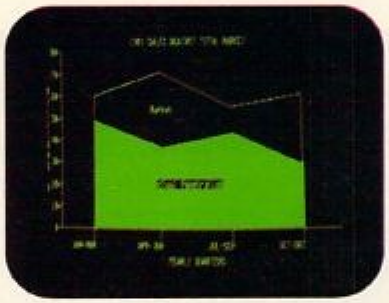


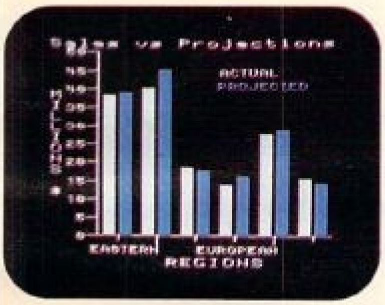



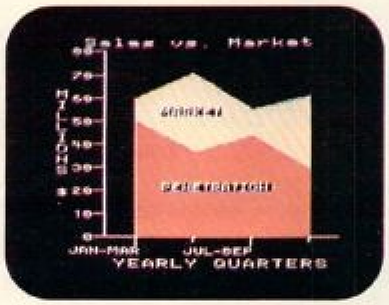



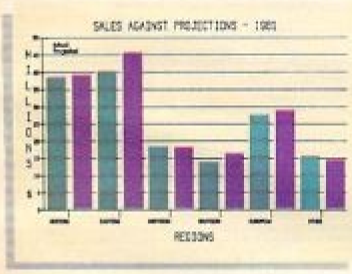
graphics are alike. pictures to prove it.

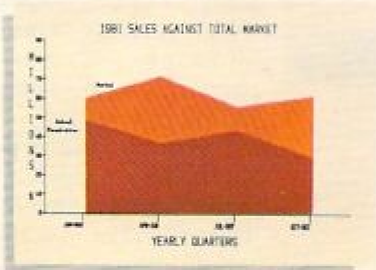


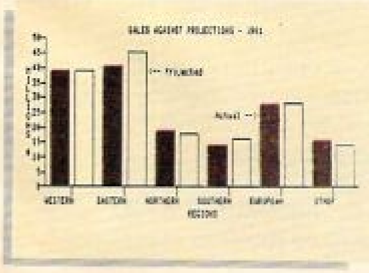


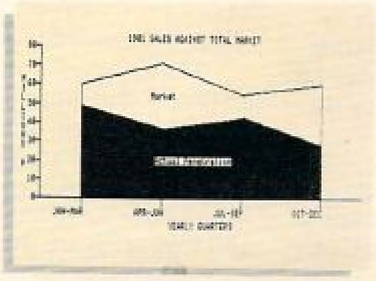










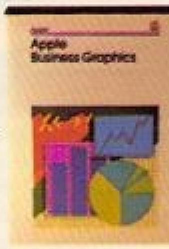


Even on transparencies and slides (by combining Apple Business Graphics with packages like Screen Director™ and Target Image Maker™).

All of which makes for more presentable presentations. And more revealing market analyses, forecasts, budgets, stock trends, business plans or customer demographics.

Or the information of your choice from the files of your choice. Be it VisiCalc®, Pascal, DIF™ or BASIC.

We could easily tell you more. But we'd rather show you more. In person. At any of our over 1300 full-support dealers (they also offer a vast library of other quality software distributed by Apple for Apples).



So pay one a visit. And find out how easy it is to turn a sea of data into data you can see.

apple®

The most personal software.

Call **(800) 538-9696** for the location of the authorized Apple dealer nearest you, or for information regarding corporate purchases through our National Account Program. Or write Apple Computer Inc., Advertising and Promotion Dept., 20525 Mariani Ave., Cupertino, CA 95014. Screen Director is a trademark of Business Professional Software, Inc. Target Image Maker is a trademark of Comshare Target Software. DIF is a trademark of Software Arts, Inc. VisiCalc is a registered trademark of VisiCorp. Pfs:Graph is a registered trademark of Software Publishing Corp.

Magic Spells For Apple

Sheila Cory

Magic Spells, by Apple Computer, is designed to give drill and practice in spelling. The program is divided into two parts: (1) a word-unscrambling game, and (2) a teacher's utility which allows the creation of word lists.

The game is set in the Castle of Spells. Merlapple, the Wizard of Spells, guides the player into the castle, where there are treasures which the player has a chance to win. Unfortunately, though, a demon also lurks in the castle, and he has his eyes on the gold. To play the game, the child first selects the word list he or she wants to be drilled on. The words are listed out, and the child is instructed either to copy the words or to just pay close attention as they're listed. The 12 word lists included on the diskette each contain 16 words.

When the player is finished looking at the words, they disappear, and one closed treasure chest appears on the screen for each word on the list. Then one of the spelling words appears on the screen, with the letters scrambled. The child is supposed to unscramble the letters to spell out one of the words that was on the list. If the word is typed in correctly, the treasure chest pops open and the gold is revealed.

The scorekeeping part of the screen displays the player's and the demon's points. Two points are accrued for each letter in the word. If the child takes more than one try to spell the word correctly, points are split with the demon. In this case, a demon holding a money bag takes the place of the closed treasure chest. If the child has to consult the demon for help (by typing an asterisk), the demon gets all points, and an empty treasure chest appears on the screen.

This procedure continues for each of the words on the list. The order of the words is random, as is the scrambling of the letters. At the end of the game, the child receives a reward based on his or her score. A typical reward is getting to make a wish on a beautiful unicorn.

Create Word Lists

The second part of this program, the teacher utility, allows the creation of word lists that are appropriate for each class setting. A menu for the teacher utility specifies the options, which include entering a list of words, deleting a list of words, viewing a list of words, seeing a list of file names, and copying a list on the printer.

Up to 20 words can be entered in each list. This will accommodate most weekly spelling lists found in elementary school spelling books. The lists will be automatically saved on diskette, unless the disk drive contains the *Magic Spells* master disk, which comes write-protected. The correct diskette to have in the drive is the *Magic Spells* backup, which is not write-protected.

My "kid consultants" for this program truly enjoyed the game. They found unscrambling the words a lot of fun. Lisa, seven years old, struggled to unscramble each word, and then never looked at the scrambled letters as she typed in the letters to spell the word correctly. It took her about a half-hour to get through one word list, but she wanted to try another when she was through. She conscientiously wrote down the words in the list when they appeared at the start of the game.

Chrissa, eight years old, did not want to ask the demon for help, preferring to wangle the help out of other people in the room. She felt that the game was hard, but enjoyed it enough to want to borrow the diskette. Cara, ten years old, played the game for about 45 minutes, and said she didn't really think she'd like to use it again.

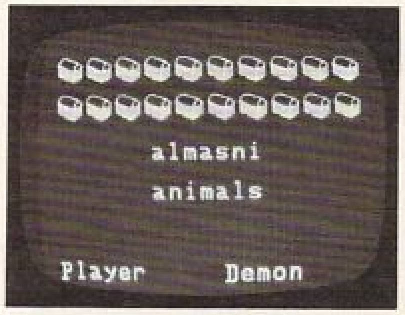
Suggested Revisions

The program seemed to pass the kid test. However, I feel that it fails to do what it is designed to do – give drill and practice in spelling. Spelling is a recall rather than a recognition task. When we try to spell a word, it is rare indeed for us to have the letters there in front of us in scrambled form. We need to pull the letters from memory. This program does not give such practice.

One way recall can be practiced would be through a tape recorder link-up with the computer so that a child would actually hear the words. A second way would be for each word to flash on the screen for a long enough time to be read, and then be erased before the child begins typing it in. A third possibility would be for the program to generate sentences with the target word omitted.

Some of the word lists on the diskette contain words that children misspell because they choose the wrong homonym. Words like "there" and "their" are in the lists, as are "to", "two", and "too". Avoiding these misspellings would be helped only with practice using them in sentences. Unscrambling their letters is of no value whatsoever.

A relatively minor problem with the program is the length of the game. Each game could take up to 40 minutes. This is longer than the amount of time children typically have on a computer at school. If a child needs



Unscrambling spelling words in Magic Spells.

to stop playing before the game is over, the ESC key will allow an escape from the game. However, when this option is exercised, no score summary is presented, and no reward is given.

The documentation for this diskette gives information about the game in an interesting, clear fashion. Screen photographs add a lot to its understandability. It gets a little confusing when it gives instructions for making backup copies of lists of words, however. The major confusion is whether the manual is referring to the *Magic Spells* backup diskette or another backup diskette. A teacher or parent new to the computer would probably need some help understanding to which disk the instructions are referring.

This program has potential, but I feel it should be revised. Although the kids love the game, the teacher utility works well, and the flow of the program is appealing and makes logical sense, it should be reworked to include an educationally sound way of giving practice in spelling words.

The program, written by Leslie Grimm, comes with a master and a backup and runs on an Apple II Plus with 48K.

Magic Spells
Apple Computer, Inc.
20525 Mariani Avenue
Cupertino, CA 95014
$45

MIDNIGHT-OIL

WE MARKET EXCEPTIONAL HOME DEVELOPED SOFTWARE!!

For a free catalog of available software or for marketing agreement information write:
MIDNIGHT-OIL
P.O. Box 1808 Beaverton, Oregon 97075

Input Into Apple's EXEC

Wally Hubbard

This simulation of the INPUT command, written in Applesoft BASIC, can be used to make EXEC files take input from the Apple keyboard.

Normally, the command INPUT A$ in an EXEC file ignores the keyboard and uses the next line in the EXEC file as its input. As an example, the file

```
INPUT A$
INPUT B$
PRINT A$, B$
```

would set A$ = "INPUT B$".

Program 1 shows a text file, EXPUT, which issues a prompt and then puts the response from the keyboard into XX$. The second line then RUNs the file named by XX$. Program 2 shows an Applesoft BASIC program which can be used to make EXEC files. It could be used to enter Program 1, but because EXPUT is so long, Program 2 contains a subroutine that automatically enters EXPUT whenever you type CTRL-I.

Let me explain how EXPUT works. It uses two FOR/NEXT loops as WHILE-WENDs, which are not explicitly available in Applesoft. The FOR/NEXT loop using B keeps cycling until B = 1. B does not equal 1 until a key has been pressed. The statement B = (X>127) sets B = 1 if the statement in parentheses is true, otherwise B = 0. And X, the value at the keyboard port, is always less than 128 until a key is pressed. The B loop gets each character, and the A loop, which is around it, puts each character into XX$ until RETURN is pressed. The sequence from FLASH to NORMAL puts the flashing cursor on the screen. The segment

```
XX$ = LEFT$(XX$,LEN(XX$)-(X = 13)-2*(X = 8))
```

subtracts one character from the end of XX$ if that last character is a carriage return [CHR$(13)], two characters if it is a backspace [CHR$(8)].

If a one-character response is all that is needed, you can simulate a GET command by eliminating the segments that affect XX$ and the statements that refer to A, including the last NEXT. This will put the character in X$.

EXPUT allows use of the left-arrow (BACKSPACE) key but does not allow use of the right-arrow or ESCape functions. A RUN, LOAD, CLEAR, or NEW command will erase the contents of XX$ and the other variables.

Using Make Exec

"Make Exec" (Program 2) is a simple, general-purpose text-entry program. The familiar Apple editing features (right-arrow, left-arrow, and pure cursor moves via the ESCape key) are available. Tap the space bar twice instead of once to get out of the ESCape functions. To back up to a previous line, type CTRL-B. To go forward one line, without changing the contents of the current line, enter a RETURN as the first character on the line. When you have finished entering all of the text, enter a ! as the first character on a new line; you will be prompted for the name the file is to be saved under. If you want to resume editing, don't enter a file name, just press RETURN. If you want to exit the program, type CTRL-C.

Most of EXPUT is automatically entered on the current line when you type CTRL-I. You must designate the contents of PR$, which is used as the prompt, and if desired, use HOME and VTAB before typing CTRL-I. Keep in mind that EXPUT is long, and lines cannot exceed 255 characters. To eliminate a chance of syntax errors, EXPUT begins with a colon.

Program 1.

```
  THE FILE 'EXPUT' CONSISTS OF TWO
LINES.  THEY ARE BROKEN INTO SEGMENTS
IN THIS LISTING FOR CLARITY.
  THE FIRST LINE GIVES THE PROMPT AND
TAKES THE INPUT.  THE SECOND EXECUTES
A COMMAND USING THE INPUT AS A
PARAMETER.  (IF THE FOR-NEXT
LOOPS ARE NOT ON THE SAME LINE THEY
WILL NOT BE EXECUTED.)

XX$="":
HOME:
VTAB 15:
?"ENTER FILE TO BE RUN: ";:
   FOR A = 0 TO 1:
   FLASH:
   ?" ";CHR$(8);:
   NORMAL:
   POKE-16368,0:
      FOR B = 0 TO 1:
      X=PEEK(-16384):
      B=(X>127):
      NEXT:
   X=X-128:
   X$=CHR$(X):
    ?X$;:
    XX$=XX$+X$:
    A=(X=13):
    XX$=LEFT$(XX$,LEN(XX$)-(X=13)-2*(X=8)):
    NEXT
PRINT CHR$(4);"RUN ";XX$
```

Program 2.

```
110  VTAB 1: INVERSE : INPUT "CLEAR SCREEN?
     (Y/N) ";A$: NORMAL : IF  LEFT$ (A$,1) =
     "Y" THEN  HOME
120  VTAB 5
130  DIM C$(100)
140  GOTO 320
150  REM   GET EACH LETTER
160  GET A$: PRINT A$;
170  IF A$ =  CHR$ (13) AND  LEN (B$) = 0 THEN
     A = A + 1: CALL  - 958: PRINT : PRINT A
     ;" ";C$(A);: FOR B = 0 TO  LEN (C$(A)):
      PRINT  CHR$ (8);: NEXT : PRINT " ";:
     GOTO 160: REM  GO FORWARD ONE LINE
180  IF A$ =  CHR$ (13) THEN  CALL  - 958:
     GOTO 300: REM  RETURN
190  IF A$ =  CHR$ (8) AND  LEN (B$) < 2 THEN
     B$ = "": GOTO 160: REM BACKSPACE IF LEN
     (B$) <= 1
200  IF A$ =  CHR$ (8) THEN B$ =  LEFT$ (B$,
      LEN (B$) - 1): GOTO 160: REM  BACKSPAC
     E IF LEN(B$)<>1
210  IF A$ =  CHR$ (21) THEN A$ =  CHR$ ( PEEK
     ( PEEK (40) + 256 *  PEEK (41) +  PEEK
     (36))): PRINT A$;: REM  RIGHT-ARROW
220  IF A$ =  CHR$ (27) THEN  CALL  - 721:
     GOTO 160: REM ESCAPE
230  IF A$ =  CHR$ (2) THEN A = A - 1:B$ = C
     $(A): CALL  - 958: PRINT : PRINT A;" ";
     B$;: FOR B = 0 TO  LEN (B$): PRINT  CHR$
     (8);: NEXT : PRINT " ";:B$ = "": GOTO 1
     60: REM   BACK UP  ONE LINE
240  IF A$ =  CHR$ (3) THEN  STOP : GOTO 160
     : REM  CTRL-C
250  IF A$ = "!" THEN 340
260  IF A$ =  CHR$ (9) THEN 500: REM  CTRL-I
270  B$ = B$ + A$
280  GOTO 160
290  REM   STORE A LINE
300 C$(A) = B$
310 B$ = ""
320 A = A + 1: PRINT : PRINT A;" ";
330  GOTO 160
340  REM   SAVE IT ALL
350 D$ =  CHR$ (4)
360  PRINT : PRINT
370  INPUT "WHAT IS THE FILE'S NAME? ";FL$
380  IF FL$ = "" THEN 160: REM   NULL
390  PRINT : PRINT "SAVING ";FL$
400  PRINT D$;"OPEN";FL$
410  PRINT D$;"DELETE";FL$
420  PRINT D$;"OPEN";FL$
430  PRINT D$;"WRITE";FL$
440  FOR B = 1 TO A
450  PRINT C$(B)
460  NEXT
470  PRINT D$;"CLOSE";FL$
480  END
490  REM   CTRL-I CALLS EXPUT
500 A$ = ":XX$=" +  CHR$ (34) +  CHR$ (34) +
     ";?PR$;:FOR A=0 TO 1:FLASH:?CHR$(32);CH
     R$(8);:NORMAL:POKE-16368,0:FOR B=0 TO 1
     :X=PEEK(-16384):"
510 A$ = A$ + "B=(X>127):NEXT:X=X-128:X$=CHR
     $(X):?X$;:XX$=XX$+X$:A=(X=13):XX$=LEFT$
     (XX$,LEN(XX$)-(X=13)-2*(X=8)):NEXT"
520  PRINT A$;
530 B$ = B$ + A$
540  GOTO 160
```

©

CRAM UP TO 90K IN YOUR ATARI

ACA introduces the next generation of memory upgrades. By using simple yet innovative techniques, the cRAM board puts complete control of up to 90K of addressable memory at your fingertips.

Features:

Powers up with 52K of continuous RAM. 4K overwrite protected RAM for assembly object code and BASIC user functions; 52K for boot users.

Cartridge ROM can be replaced by RAM under program control. Simplifies cartridge program development and testing; or 8K/16K hidden RAM for temporary data.

OS ROMs can be replaced by RAM, for OS development or use of an alternate OS.

Built-in flexibility. Several customizing options have been built into the cRAM board. These, along with the features above, convert your Atari 400 into a truly powerful tool.

IN STOCK for immediate shipment.

INTRODUCTORY PRICE

KIT: $169.00
(regular price $189.00)

ASSEMBLED AND TESTED: $209.00
(regular price $229.00)

ACA

Applied Computer Alternatives, Inc.

1600 Wilson Boulevard, Suite 900
Arlington, Virginia 22209
(703) 525-6960

DEALER INQUIRIES INVITED.

Requires minor modification to your Atari RAM board. For Atari 400 only.

ATARI 400 is a registered trademark of ATARI, Inc.

Apple Subroutine Capture

R. W. W. Taylor and Max Hailperin

Do you include certain favorite BASIC subroutines in program after program? The easiest way to incorporate a standard subroutine into a new program is to EXEC the code from an existing text file, as explained on page 76 of the *Apple II DOS Manual*. A short program is given on that page for "capturing" specified lines from a program already in memory and writing the lines as text to a sequential file for later retrieval by an EXEC command.

The main inconvenience of this particular approach is that the capture subroutine must be typed in new each time it is to be used, with details specific to the situation at hand.

This nuisance can be avoided. In fact, it is possible to create and store a master file Capture so that a user who simply types EXEC CAPTURE will be interrogated about the desired file name and line-number range, and the desired capture will then be performed without any further action by the user.

The text of Capture appears in Program 1. This text can be entered into a file by a program such as File Builder (Program 2). Note the subroutine at line 8000. The purpose of this subroutine is to allow input of arbitrary text strings, including commas, colons, and hyphens. It is a good example of the sort of subroutine that is handy to capture and maintain for re-use in other programs.

Saving To Memory

Once Capture has been stored on disk, and a program containing lines to be captured has been loaded or created, the command EXEC CAPTURE is issued. The first effect is to overlay lines 1-18 of the program in memory – lines in this range cannot be captured. These lines are then run by the RUN at the end of Capture. The user is asked to specify a name for the file to be created and two line numbers indicating the range of code to be captured. The line numbers must be entered separated by a *comma*.

The program then proceeds to build a file called Tempcapture, incorporating the information supplied by the user. Before ending, the program issues a command to EXEC TEMPCAPTURE. Once again, lines in the range 1-18 are overlaid, and the new lines are run. This time, the desired capture is performed, Tempcapture is deleted, and the completion of the task is announced.

Note that if the user's disk already happens to contain a text file named Tempcapture, this file will be overwritten and then deleted. An already existing text file will also be overwritten if its name is specified as the file to be created. However, if the name specified represents an existing binary, Applesoft, or integer file, a "FILE TYPE MISMATCH" message will be generated, and the process will halt without any damage to the file.

Program 1: Text For Capture File

```
1  REM  - CAPTURE SUBROUTINE
2 CD$ =  CHR$ (4): REM   CONTROL D
3 Q$ =  CHR$ (162): REM   QUOTE CHARACTER
4  HOME : INPUT "FILE NAME TO BE CREATED? ";F$
5  VTAB 4: INPUT "LINES TO BE CAPTURED? ";LO
   %,L1%
6  PRINT CD$;"OPEN TEMPCAPTURE"
7  PRINT CD$;"WRITE TEMPCAPTURE"
8  PRINT "4 PRINT CD$;";Q$;"OPEN ";F$ + Q$
9  PRINT "5 PRINT CD$;";Q$;"WRITE ";F$ + Q$
10  PRINT "6 LIST ";LO%;"-";L1%
11  PRINT "7 PRINT CD$;";Q$;"CLOSE ";F$ + Q$
12  PRINT "8 PRINT CD$;";Q$;"DELETE TEMPCAPT
    URE";Q$
13  PRINT "9 HOME: PRINT ";Q$;"FILE ";F$;" H
    AS BEEN CREATED.";Q$
14  PRINT "10 END"
15  PRINT "RUN"
16  PRINT CD$;"CLOSE TEMPCAPTURE"
17  PRINT CD$;"EXEC TEMP CAPTURE"
18  END
```

Program 2: EXEC File Builder

```
10  REM  ** FILE BUILDER **
20 CD$ =  CHR$ (4): REM  CONTROL D
30  HOME : PRINT "ENTER NAME OF FILE TO BE B
    UILT:"
40  PRINT : HTAB 10: INPUT F$: HTAB 10: VTAB
     PEEK (37): PRINT " "
50  PRINT : PRINT "INPUT LINES ONE BY ONE."
60  PRINT "TO END, JUST PRESS RETURN."
70  VTAB 9: POKE 34,8: REM   SET TOP OF TEXT
     WINDOW
80  PRINT CD$;"OPEN ";F$: PRINT CD$;"DELETE
    ";F$: PRINT CD$;"OPEN ";F$
90  FOR I = 0 TO 1
100  PRINT "* ";: GOSUB 8000
110  IF 0 <  LEN (IN$) THEN I = 0: PRINT CD$
    ;"WRITE ";F$: PRINT IN$: PRINT CD$
120  NEXT I
130  PRINT CD$;"CLOSE";F$
140  HOME : POKE 34,0: REM   RESET TEXT WINDOW
150  PRINT "* FILE ";F$;" HAS BEEN BUILT."
160  END
8000  CALL 54572: REM  INPUT SUBROUTINE
8010  FOR B = 512 TO 751
8020  IF  PEEK (B) <  > 0 THEN  NEXT
8030 IN$ = ""
8040  POKE  PEEK (131) + 256 *  PEEK (132) +
    1,0
8050  POKE  PEEK (131) + 256 *  PEEK (132) +
    2,2
8060  POKE  PEEK (131) + 256 *  PEEK (132),B
     - 512
8070 IN$ =  MID$ (IN$,1)
8080  RETURN
```

©

Custom Catalog

G. J. Vullings

For Apples with DOS 3.2.1 or 3.3, all memory sizes, this program lets you create customized directory headers, with inverse or normal input.

Have you ever wished to personalize your disks, identify the theme of a series of programs on a disk, or just improve the appearance of the directory as it appears on the screen after a CATALOG command? "Custom Catalog" will allow you these prerogatives and more by creating seven "bogus" files at the top of the directory. These bogus files will serve as a header to the disk's directory, displaying contents, ownership, DOS version, or whatever you wish.

The program is designed to run in either a DOS 3.2.1 or 3.3 environment. It will, additionally, permit either inverse or normal input and will allow toggling between the two input states. These features make possible directories with content and artistry.

Your Choice Of Input Types

The program should be used only with newly-initialized diskettes since it will occupy the first seven entries in the directory. Thus, if the program is used with established diskettes, the first seven programs will become inaccessible. To implement Custom Catalog, initialize a diskette the normal way and then delete the "HELLO" program. Run Custom Catalog and, when prompted, insert the diskette to be customized.

You have an initial choice of input states (normal or inverse) and may then design seven lines of 23 characters each (any except for control characters) to represent your identifying remarks or messages. The program sets aside a buffer of 256 bytes in high memory, using the input/output block at decimal location 896. There it stores the last sector of the directory track, which is normally track 17, sector 12 or sector 15, depending on the DOS version.

Each directory entry occupies 35 bytes. The first two represent the track and sector of the track/sector list (header). These we will direct towards an empty sector, namely track 17 (in most cases), sector one. The third byte represents the file type. Here we will use "00" to indicate an unlocked text file. The next 30 bytes represent the file name. We will make the first seven bytes backspaces to eliminate the "t" (for text) and the sector count from the display. The remaining 23 bytes may be anything of your choosing (normal or inverse). The 34th byte is the file length. This we set to "00" and the last byte is the end marker, which is also normally "00".

We now alter the entries in the buffer, but one problem remains. The output for the directory listing is via the COUT routine at $FDED using screen ASCII values, but keyboard ASCII values which we entered are in a different range. We can translate these values listing logical variables (one of the least used, yet very powerful, variable types). See page 15 of the *Apple Reference Manual* for screen ASCII values for both normal and inverse display. After altering the buffer, the revised version is written back to the disk.

Using a similar technique, track 2, sector 2 is then read into the buffer and the DISK VOLUME message which occupies the 176th to 186th bytes, inclusive, may be optionally changed.

A typical directory header might look like the following example:

```
&&&&&&&&&&&&&&&&&&&&&&&&
& APPLE II - DOS 3.3   &
&    DISK UTILITIES    &
&     JANUARY 1983     &
&PROP. OF G.J.VULLINGS&
&&&&&&&&&&&&&&&&&&&&&&&&
```

The above looks especially attractive in inverse mode. A hint: if the seventh line is left blank, a natural break is formed to separate the header from the rest of the directory.

Both backspace and forwardspace editing are implemented and work normally, with a few enhancements (because of the two modes of input). The only exception to normal implementation is that you cannot backspace beyond the first column (column numbers are provided to make centering easier). Therefore, pressing "RETURN" or typing past column 23 is final.

After the seventh line is entered, you are given the choice of accepting or rejecting the header that you have constructed. If you reject it, the procedure will begin again. Rejecting headers will give practice in obtaining a result which is aesthetically pleasing. If, on the other hand, you accept the header, it will be permanently written to the disk. The choice will then be offered to change the DISK VOLUME message to any 11 (or fewer) characters of your choice.

You can create additional custom entries using the method in this program, or you might want flashing entries, which you can get by translating to the required ASCII values. Have fun experimenting, and happy customizing.

How It Works

LINES

30-220 – the input routine, which allows input in two modes as well as forwardspace and backspace editing.

250-260 – translate keyboard ASCII into screen ASCII and store into disk buffer.

280-290 – toggle input status.

310-330 – backspace edit routine.

350-390 – forwardspace edit routine. Translate screen ASCII to keyboard ASCII.

410-450 – point each of the "bogus" header files to empty track 17, sector 1; declare each file to be of type "text-unlocked" of length zero; and set the end marker.

470 – inputs a series of seven backspaces into the filenames so that the lock indicator, file type, and sector count do not appear on screen.

480 – checks the memory size of your Apple and sets up a disk buffer, making the program virtually memory-size independent.

500-570 – organize screen display.

590-620 – set HIMEM: to protect the buffer and also initialize the variables.

640-670 – use track 17, sector 0, to find the directory, thus making it possible to use the program with either DOS 3.2.1 or 3.3, or even with disks having directories on tracks other than track 17.

680-800 – main routine.

820-840 – write the catalog header to the disk.

860-920 – change DISK VOLUME message.

940-990 – finishing touches.

1020-1040 – set up the input/output block for the Read Write Track Sector routine.

```
5  TEXT : HOME : ONERR  GOTO 1000
10  GOTO 480
20  REM   ***.INPUT ROUTINE.***
30  FOR I = 0 TO 6
40  VTAB VTB + I: HTAB HTB
50 CN = 1
60  INVERSE
70  IF  NOT INV THEN  NORMAL
80  GET CH$: IF CH$ <  >  CHR$ (13) THEN 110

90  IF CN > 23 THEN 200
100  FOR Z = CN TO 23:CH$ = " ": PRINT CH$;:
     GOSUB 250:CN = CN + 1: NEXT : GOTO 200

110  IF CH$ =  CHR$ (27) THEN  GOSUB 270: GOTO
     60
120  IF CH$ =  CHR$ (8) THEN  GOSUB 300: GOTO
     60
130  IF CN > 23 THEN 200
140  IF CH$ =  CHR$ (21) THEN  GOSUB 340: GOTO
     160
150  IF  ASC (CH$) < 32 THEN 60
160  PRINT CH$;
170  GOSUB 250
180 CN = CN + 1
190  GOTO 60
200  GOSUB 460
210  GOSUB 400
220  NEXT
230  RETURN
740  REM   ***.SCRN ASC INTO BUFFER.***
250  IF  ASC (CH$)  >  = 32 AND  ASC (CH$) <
     64 THEN  POKE BFR + I * 35 + 10 + CN, ASC
     (CH$) + ( NOT INV > 0) * 128: RETURN
260  POKE BFR + I * 35 + 10 + CN, ASC (CH$) -
     (INV > 0) * 64 + ( NOT INV > 0) * 128: RETURN
270  REM   ***.CHANGE INPUT STATE.***
280  IF INV THEN INV = 0: RETURN
290 INV = 1: RETURN
300  REM   ***.BACKSPACE ROUTINE.***
310 CN = CN - 1: IF CN = 0 THEN  POP : GOTO
     50
320  PRINT CH$;
330  RETURN
340  REM   ***.FORWARD SPACE ROUTINE.***
350 ASKII =  PEEK ( PEEK (40) + 256 *  PEEK
     (41) +  PEEK (36))
360  IF ASKII < 32 THEN CH$ =  CHR$ (ASKII +
     64): RETURN
370  IF ASKII < 64 THEN CH$ =  CHR$ (ASKII):
      RETURN
390 CH$ =  CHR$ (ASKII - 128): RETURN
400  REM   ***.PLACE COMMON POINTERS.***

410  POKE BFR + I * 35 + 1,TRK
420  POKE BFR + I * 35 + 2,1
430  POKE BFR + I * 35 + 3,0
440  POKE BFR + I * 35 + 34,0
450  POKE BFR + I * 35 + 35,0: RETURN
460  REM   ***.PUT BKSPACES IN DIRECTORY.***

470  FOR M = 4 TO 10: POKE BFR + I * 35 + M,
     136: NEXT : RETURN
475  REM   ***.SET DISK BUFFER.***
480 BL =  PEEK (115):BH =  PEEK (116) - 1:BU
     FR = BL + BH * 256
490  REM    ***.INITIALIZE SCREEN.***
500  TEXT : HOME : VTAB 2: INVERSE : FOR I =
     1 TO 40: PRINT "=";: NEXT
510  PRINT "=       APPLE II CATALOG CUSTOMIZ
     ER      =";
520  PRINT "=";
530  FOR I = 1 TO 38: PRINT " ";: NEXT
540  PRINT "=";
550  PRINT "=                  ";: NORMAL : PRINT
     "BY G.J. VULLINGS";: INVERSE : PRINT "
           =";
560  FOR I = 1 TO 40: PRINT "=";: NEXT : NORMAL

570  POKE 34,6
580  REM   ***.INITIALIZE VARIABLES.***
590  HIMEM: BUFR:IOB = 904:ITRK = IOB + 4:IS
     ECT = IOB + 5:IBUFP = IOB + 8:ICMD = IO
     B + 12:ST = IOB + 13:RWTS = 896:D$ =  CHR$
     (13) +  CHR$ (4):RD = 1:WRT = 2:BFR = B
     UFR + 10
600  GOSUB 1020: POKE IBUFP,BL: POKE IBUFP +
     1,BH
610  HOME : VTAB 20: PRINT "INSERT DISK TO B
     E CUSTOMIZED"
620  VTAB 22: PRINT "THEN PRESS ";: INVERSE
     : PRINT " RETURN ";: NORMAL : GET Z$: PRINT
     Z$
630  REM   ***.READ CATALOG INTO BUFFER.***

640 TRK = 17:SECTR = 0
650  POKE ITRK,TRK: POKE ISECT,SECTR: POKE I
     CMD,RD: CALL RWTS
660 TRK =  PEEK (BUFR + 1):SECTR =  PEEK (BU
     FR + 2)
670  POKE ITRK,TRK: POKE ISECT,SECTR: CALL R
     WTS
```

```
675  REM   ***.MAIN ROUTINE.***
680  HOME : VTAB 20: PRINT "(I)NVERSE OR (N)
     ORMAL ";: GET A$: PRINT A$:VTB = 12:HTB
      = 8
690 INV = 0
700  IF A$ = "I" THEN INV = 1
710 Z1$ = "00000000011111111112222"
720 Z2$ = "12345678901234567890123"
730  VTAB 10: HTAB HTB: PRINT Z1$: HTAB HTB:
      PRINT Z2$
740 TB = 12: FOR Z = 0 TO 6
750  VTAB TB + Z: HTAB 7: PRINT "+";: IF INV
      THEN  INVERSE
760  FOR J = 1 TO 23: PRINT " ";: NEXT : NORMAL
     : PRINT "+"
770  NEXT
780  VTAB 20: CALL  - 958: HTAB 5: PRINT "IN
     PUT LINES OF CUSTOM CATALOG"
790  VTAB 22: PRINT " PRESS ";: INVERSE : PRINT
     " ESC ";: NORMAL : PRINT " TO CHANGE DI
     SPLAY STATUS"
800  GOSUB 30: NORMAL
810  REM   ***.WRITE SECTOR TO DISK.***
820  PRINT : VTAB 20: CALL  - 958: PRINT "IS
      THIS WHAT YOU WANT? (Y/N) ";: GET ZZ$:
      PRINT ZZ$
830  IF ZZ$ = "N" THEN 680
840  POKE ICMD,WR: CALL RWTS
850  REM   ***.CHANGE DISK VOLUME.***
860  PRINT : PRINT "IS ";: INVERSE : PRINT "
      DISK VOLUME ";:: NORMAL : PRINT " TO B
     E REPLACED? (Y/N) ";: GET Z$: PRINT Z$
870  IF Z$ <  > "Y" THEN 930
880 TRK = 2:SECTR = 2: POKE ITRK,TRK: POKE I
     SECT,SECTR: POKE ICMD,RD: CALL RWTS
890  INPUT "INPUT 11 CHARACTER HEADER: ";MS$
     :LN =  LEN (MS$): IF LN >  = 11 THEN MS
     $ =  LEFT$ (MS$,11): GOTO 910
900  FOR I = LN + 1 TO 11:MS$ = MS$ + " ": NEXT
910 J = 0: FOR I = BUFR + 176 TO BUFR + 186:
      POKE I, ASC ( MID$ (MS$,11 - J,1)) + 1
     28:J = J + 1: NEXT
920  POKE ICMD,WR: CALL RWTS
930  REM   ***.DISPLAY CATALOG AND FINISH.**
     *
940  HOME : PRINT D$"CATALOGD1"
950  PRINT : PRINT "MORE CUSTOMIZING? (Y/N)
     ";: GET ZZ$: PRINT ZZ$
960  IF ZZ$ = "Y" THEN 610
970  TEXT : HOME : VTAB 10: HTAB 11: FLASH :
      PRINT " SEE YA' LATER!! ": NORMAL
980  VTAB 23: END
990  RETURN
1000  HOME : PRINT "***.ERROR.***": END
1010  REM   ***.SET-UP IOB.***
1020  FOR I = 1 TO 25: READ I%: POKE 896 + I
      - 1,I%: NEXT I: RETURN
1030  DATA  160,136,169,3,32,181,183,96,1,96
     ,1,0,17,15,251,183,0,128,0,0
1040  DATA  2,2,254,96,1,59,236,236,59,59,23
     6,236,59,27,236,28,29,30,236,236
```

©

COMPU SENSE

CENTIPOD **$27.95**
Like Centiped, only better!

FROGEE **$27.95**
The exciting arcade game of Frogger.

MOTOR MOUSE **$29.95**
What a cheese'ee game!

CRIBBAGE
VIC-20 **$14.95** C-64 **$17.95**
This is the game of Cribbage.

STAR TREK
VIC-20 **$12.95** C-64 **$17.95**
Excellent adventure game!

MASTER MIND
VIC-20 **$12.95** C-64 **$19.95**
Makes you think.

ROACH MOTEL **$9.95**
Kill the bugs!

YAHTZEE 1.1 **$12.95**
YAHTZEE 2.1 **$14.95**

GENERAL LEDGER **$19.95**
(VIC-20)

CHECK MINDER
VIC-20 **$19.95** C-64 **$24.95**

HOME INVENTORY **$19.95**
(VIC-20)

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

VISA
MasterCard

Personal checks accepted
(Allow 3 weeks) or
C.O.D. (Add $2.00)
Handling charges $2.00
VIC-20® is a registered trademark of Commodore

commodore VIC-20™ Computer

VIC's MOM — MOTHER BOARD EXPANDER

Expand your vic to full limit	
3 slot	$39.95
provisions for switches in board	
with switches	$45.95
CHARACTER BUILDER — UTILITY AID	
will save and load from disk or tape; design your own CUSTOM CHARACTERS; use with any memory configuration	$25.00
RS-232 bi-directional INCLUDES SECOND JOY STICK PORT	$40.00
"NEW" SECOND JOY STICK PORT	$20.00
VIC-20 to EEE-488 Interface	
allows VIC to use PET/CBM Peripheral	$79.95
ROM EXPANSION BOARD	
put your own programs on 2K or 4K EPROM'S; we can put your program in ROM— call for info	$19.50
8K RAM BOARD can be daisy chained to four	$49.95
4K RAM BOARD	$39.95
VIC DUST COVER protect your VIC	$12.95

NEW CARTRIDGE VIDEO GAMES FROM MACHINE
LANGUAGE, INC. — SUPER FAST**COLORFUL
AVAILABLE THRU OEM, INC. $25.00

TWO PLAYER GAMES
CLOSE ENCOUNTERS OF THE WORST KIND • BLACK JACK
ACID RAIN • BLOCK BUSTER • DOT GOBBLER • FROGMAN
SPIN TOIDS • CHES
we need good new machine language games - royalty paid
CALL FOR DETAILS
Completely Tested, 100% Guaranteed

MasterCard
VISA

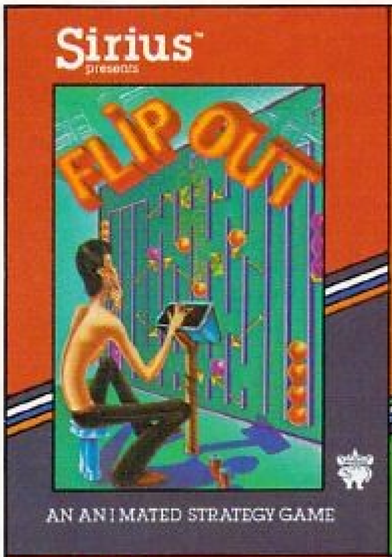
Dealer inquiries invited
TO ORDER CALL – 305-465-9363

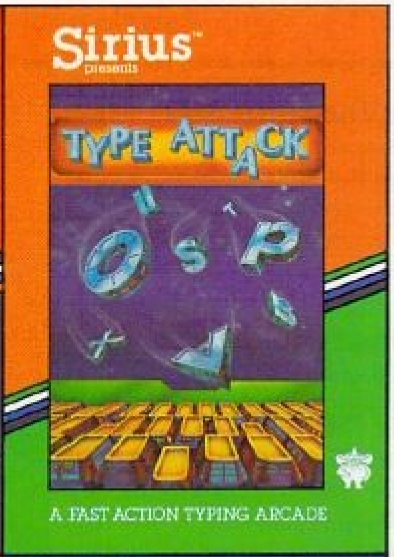
order from OEM Inc.
2729 South U.S. 1, Suite 12
Ft. Pierce, Florida 33450

Personal checks accepted allow time to clear — add $2 for shipping — add $5 for COD
Florida residents add 5% state sales tax — Master Card & Visa add 5%

Have A Great Playday!

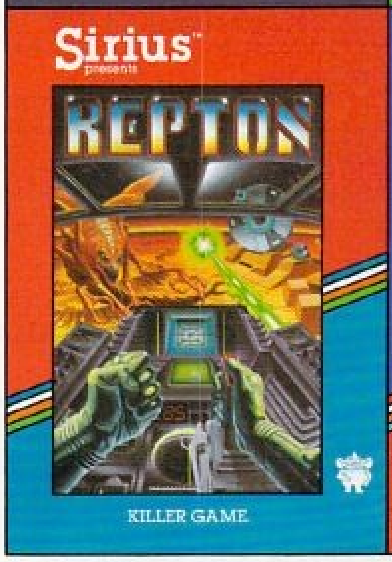
Take your marble to the top. Pick your spot and let it drop. Hope for a flip instead of a flop. Once you get it, the fun never stops! It's FLIP OUT — a crazy new strategy game for one or two players. Each marble you drop causes a chain reaction, so take your time and plan carefully. Plan right and you'll flip, if you didn't you Flip Out!

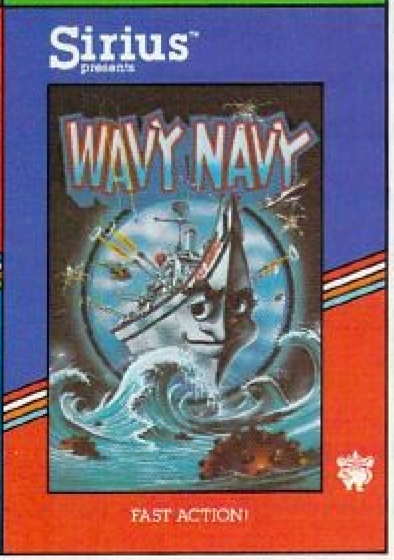




Turn your keyboard into a typing arcade! You can blast attacking letters and words right out of the sky. Type Attack was designed by a professional educator and the fast action game experts at Sirius. It features 39 preprogrammed lessons and 60 user defined lessons. Great sound, graphics and a realtime words per minute bar make improving your typing skills fun!

It is up to you to stop the invasion of the evil Quarriors and save Repton. You are armed with devastating Nuke Bombs, a Radar Screen, a Laser Gun and an Energy Shield. You'll need them all! You'll be attacked by Nova Cruisers and Single Saucers. You must avoid Spye Satellites and deadly Dyne-Beam Shooters and you must stop the Draynes from depleting the Reptonian power supply. Repton is a battle so thrilling you'll be relieved to find out you're still on earth when it's over!





Talk about adventure on the high seas! You're blasting away at a squadron of enemy bombers and Kamikaze fighters from the deck of your P.T. boat. Suddenly you notice the sea is loaded with mines and an Exocet missile is screaming toward you on the horizon. Instinctively you jerk the joystick to the starboard, keeping your thumb on the fire button. Phew! That was close! Sometimes it's hard to believe Wavy Navy's just a video game.

New Games For Your Apple II From Sirius™

For more information contact your local Sirius dealer or distributor or contact us at 10364 Rockingham Drive, Sacramento, CA 95827, (916) 366-1195.

Turbocharger For Apple

Richard Cornelius

Turbocharger is a Disk Operating System (DOS) and date-stamping program for the Apple II written by Roland Gustafsson. The disk comes in a plastic bag with a folded sheet of heavy stock on which the documentation is printed. The disk "may be copied by the original purchaser only as necessary for use on the computer for which it was purchased," according to the instructions.

The feature of *Turbocharger* that many users would find most impressive is the increased speed for the DOS commands BLOAD, BRUN, LOAD, and RUN. The DOS in memory is changed to the fast DOS when the file TURBO is BRUN. How much are the DOS commands speeded up?

To answer that question I wrote a simple BASIC program to BLOAD a high-resolution picture (a 34-sector file containing 8K of graphic image) ten times in succession. Normal DOS ran this program in 90 seconds. The *Turbocharger* DOS completed the task in 25 seconds.

For comparison, I tried another "fast" DOS which is used by a major commercial software publisher, and the same program took 32 seconds. I also tested each of the DOS variations with an Applesoft program. Normal DOS required 22 seconds to load an Applesoft program that occupied 89 sectors on the disk. Both *Turbocharger* and the other fast DOS that I tried loaded the same program from the same disk in about five seconds. Since loading times include the start-up time for the disk drive, the actual time for loading the program was decreased roughly by a factor of five.

On the surface, the fast DOS seems to operate very well. When I began to use it in my own software development, however, I quickly encountered a problem. The two programs below show circumstances under which the *Turbocharger* DOS seems to be failing to CLOSE the DOS input/output buffers properly.

Program 1.

```
100 D$=CHR$(4)
110 PRINT D$ "BSAVE BINARY FILE,
    A768,L1"
120 PRINT D$ "BRUN TURBO"
130 PRINT D$ "MAXFILES1"
140 PRINT D$ "RUN SECOND
    PROGRAM"
```

Program 2.

```
200 REM ....................
210 REM ....................
220 REM ....................
230 REM ....................
240 REM ....................
250 REM ....................
260 D$=CHR$(4)
270 PRINT D$ "BLOAD BINARY FILE"
```

I initialized a disk with the first program and saved the second program as SECOND PROGRAM. When I booted the disk, I received a NO BUFFERS AVAILABLE error in the last line of the second program. I do not understand the source of the error. If line 120 in the first program is changed to BRUN a dummy file, no error occurs. If one of the lines 200-250 in the second program is deleted, then no error occurs! Whatever the source of the problem, it can apparently be overcome by placing a PRINT D$ "CLOSE" statement into a new line 265 in the second program. For personal use the fast DOS is probably satisfactory, but for serious developmental work, caution is in order.

Date Stamping

The other major feature available with *Turbocharger* is the "date-stamping" of files. A one-line Applesoft program that is supplied on the disk is used to change the date. Whenever you save a file, the current date is also saved. The catalog has the normal appearance except when the command MON O (a standard DOS command) is used. This command shows the date at the right-hand edge of the screen.

For long file names (DOS allows up to 30 characters) as many as seven characters at the end of the file name may be wiped off the screen by the date. RESET or the DOS command NOMON O makes the catalog appear in the normal manner. In either catalog display, the number of free sectors on the disk is given at the top.

Two other programs are available on the *Turbocharger* disk. One is a DOS command changer that allows you to change the DOS commands on a disk. The documentation says that you can change the commands to "anything that you want." The limitations that do exist (for example, the length of the commands) are not explained in the documentation, but these limitations are not ones that a user would likely encounter. The greatest value of changing the words used for the various DOS commands is generally in shortening them so that one or two characters can be used in place of the standard commands. For the purpose of changing the commands to one or two letters, the command changer program performs its task without any problems.

The other program on the disk is a "quick-copy" program. The added features of the *Turbocharger* DOS have replaced the INIT command so that you cannot initialize disks with the fast DOS. Other fast DOS programs generally operate in the same manner. You must initialize disks either by using the normal DOS or by using the COPYA program on the System Master Disk to copy a disk that is already initialized. Once you have an initialized disk, the quick-copy program will copy the contents of one disk onto the newly initialized disk.

The documentation that accompanies the *Turbocharger* disk is brief but complete. It includes a suggestion on what to include in your HELLO programs to make changing the date easy, and it presents information on which zero-page locations are used, how the date is linked to the file name, and where in DOS the altered routines lie.

Turbocharger
Silicon Valley Systems
1625 El Camino Real
Suite #4
Belmont, CA 94002
$29.95

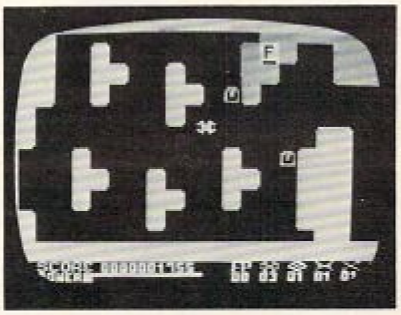
Pathfinder For Atari

John DiPrete

Programmer Randy Jongens may have taken his cue from Three Mile Island when he decided it was time for a game about radioactive materials in Gebelli Software's latest release for the Atari, *Pathfinder*.

Your Pathfinder is a "being" which moves at a velocity similar to that of *Pac-Man*'s through a maze several times larger than the viewing screen. Each time you glide over a canister of nuclear waste, you absorb energy. Hoping to out-power you is a foe called Nuke which also collects energy. Until it's strong enough to pose a threat, however, Nuke flees from you. A bodyguard called Minelayer safeguards Nuke by planting mines to block you. If your Pathfinder blasts the booby-traps, a fire ignites. The only way to douse the electro-light is to get fire-retardant pellets from a fire station. To enliven the spectacle, Phantoms zigzag through walls at you.

Pathfinder zaps the enemy at long-distance, recharging itself by gulping down "hot" spillage. A display at the bottom of the screen offers helpful data regarding power indicators, target numbers, and remaining Pathfinders. You control the waste-eater's motions by aiming the joystick in eight possible directions, pressing the button to fire plasma-blasts. With enough fire-energy, you can abolish maze walls. Angle shots, in any 45-degree direction (NE, NW, SE, SW), are not easy.


Searching for canisters of nuclear waste in the mazes of Pathfinder.

The graphics in *Pathfinder* are abstract. Squares, angles, and bric-a-brac constitute Nuke, Pathfinder, and the rest of the characters in the game. The shapes are flat, one-dimensional. No human, extra-terrestrial, or vessel-bearing features exist. No sharply-defined expressions signify the type of life (human, alien, or robotic). The rapidly-blinking geometric figures are hazy, vague, and specter-like. Maze walls remain completely solid, except at the beginner's level, where lattice-type structures exist. The instruction sheet doesn't identify the squashed-up pretzel-things that turn up now and again, so it's hard to realize at first that they're "residue" of half-crumpled targets. (A succession of plasma-blasts is required to vaporize a wall – if only a tiny dose is received, it remains in partial form.)

FOR THE WONDERFUL WORLD OF ATARI 400 & 800 SYSTEMS

RCE ANNOUNCES . . .

COMMANDER 2400

AN INVITATION TO AN EXCITING NEW DIMENSION OF COMPUTER CONTROL AND PROGRAMMING EASE.

2400 WITH KEY PAD – 2400 STANDARD

EXPERIENCE . . . . .

. . . The responsive feel of superbly crafted engineering under your fingertips.
. . . The convenience and comfort of your own detachable professional keyboard system.
. . . The beauty, elegance and natural warmth of wood.
. . . The luxuriousness, softness and durability of fine furniture textured vinyl.
. . . A combination of features designed to return the thrill of personal command to computing.

FEATURES

1. Exclusive and unique calculator circuit! Allows keypad to be switched into use as a standard rapid entry calculator.
2. Detachable option allows easy disconnection to store away while the youngsters play their games!
3. User installable in minutes, no soldering required!
4. Allows simultaneous use of BOTH keyboards!
5. Keyswitches and components are top quality design and manufacture.

THE COMMANDER 2400 IS AVAILABLE FROM $119.00 TO $219.00 DEPENDING ON YOUR CHOICE OF OPTIONS AND COMES WITH A 10 DAY MONEY BACK GUARANTEE. OUR WARRANTY IS FOR 6 MONTHS, BOTH PARTS & LABOR! SEND FOR OUR FREE BROCHURE AND FULL ATARI CATALOG!

TO ORDER, CALL TOLL FREE
(800) 547-2492

RCE

536 N.E. "E" STREET
GRANTS PASS, OREGON 97526

Applesoft Printer Control

Eric and Sally Martell

If your printer has several modes, you may have had difficulty trying to remember all the codes. This mode-setting program makes the selection of printer typefaces simpler and easier. The program is designed for the Apple II+ using an Epson printer with Graphtrax+, but it can be adapted for other printers and computers.

The dot matrix printer has evolved over the last two years from a rather stodgy machine suitable only for making nearly illegible program listings and data dumps into a sleek, glossy beast which can come close to letter quality printers in typeface formation.

The modern dot matrix printer is usually faster than letter quality printers, usually cheaper (although there is some overlap in prices), and can be the printer of choice for every application from programming to draft quality (sometimes called correspondence quality) word processing.

Many contemporary dot matrix impact printers have extensive abilities to present different styles of character formation (see the figure). Generally, these different character fonts are software selectable, a convenient feature for the user. Obviously, having the print style under program control can be useful. However, the problem then arises of remembering how to set the different print modes.

Printer manufacturers have not standardized printer control codes. Different printers will respond to different control characters. This is not a problem if your printer has only two different print fonts, but you will probably not be able to remember all of your machine's codes if it has several different printing modes. The usual answer to this problem is to look in the instruction manual which came with the printer, a solution which can be a major research project. A quicker method is to write a mode-setting program for your printer. You can do this by adapting the program presented in this article.

The program is written in Applesoft BASIC and allows the Apple II+ to use one of the Epson MX-series printers with Graphtrax+. These printers have 12 print modes which can be used in either normal or italic typeface.

Print Styles

Lines 200-340 contain all of the Epson control codes for the different type styles to be used. These lines set values into two string arrays. The array PS$(n) contains the code to set the style of type, and the array DS$(n) contains a brief description of the corresponding control code in PS$(n). These codes are specific to the Epson printers; if you're customizing this routine for another printer, you will make the majority of changes here.

The program will not only allow you to set your printer, but will also demonstrate all of the printer's various print styles if required. If you choose to print the demonstration, control passes to the routine between lines 400 and 560.

On the other hand, if you simply want to set the printer, lines 570-670 print a menu, lock it in position on the screen, and then allow input of your choice. If the printer can be set to your choice, the control passes to the short routine at line 760, which first sets the printer to the normal mode and then sends the special mode requested and returns to the menu.

The logic of this part of the program is complicated by the fact that the Epson double-width modes (modes 7-12) may be set only for a single line and must be reset at the beginning of every line to be printed in those modes. Lines 690-710 print a message about this problem, and then lines 720 and 730 allow you to type in a brief message, which will be printed on the printer as soon as you hit the return key. The printer will be left in the normal 80 characters per line mode. After printing the line, control returns to the menu routine.

It must be noted that every time a mode is sent to the printer, all previous modes are cleared by first sending PS$(3), the "normal" mode command string. The string, PS$(3), resets double strike, compressed, and enhanced modes, but does not reset the Italics command. Therefore, if you first request the special print mode which you want and then turn the Italics on, you will get the normal mode with Italics. The correct procedure: first turn on the Italics and then select the special print desired. If you then use option 15 to end the program, your printer will remain set in the typeface which you have specified.

Regardless of which brand of printer or computer you own, the basic approach used here is easily customized. With a look at your printer's manual and a little work keying in code, you should never have difficulty selecting printer typefaces again.

Figure.

```
COMPRESSED MODE
0123456789 AaBbCcDdEeFfGgHhIiJjKkLlMmNnOoPpQqRrSsTtUuVvWwXxYyZz
COMPRESSED-DOUBLE STRIKE MODE
0123456789 AaBbCcDdEeFfGgHhIiJjKkLlMmNnOoPpQqRrSsTtUuVvWwXxYyZz
NORMAL MODE
0123456789 AaBbCcDdEeFfGgHhIiJjKkLlMmNnOoPpQqRrSsTtUuVvWwXxYyZz
NORMAL DOUBLE STRIKE MODE
0123456789 AaBbCcDdEeFfGgHhIiJjKkLlMmNnOoPpQqRrSsTtUuVvWwXxYyZz
NORMAL-EMPHASIZED MODE
0123456789 AaBbCcDdEeFfGgHhIiJjKkLlMmNnOoPpQqRrSsTtUuVvWwXxYyZz
NORMAL-EMPHASIZED-DOUBLE STRIKE
0123456789 AaBbCcDdEeFfGgHhIiJjKkLlMmNnOoPpQqRrSsTtUuVvWwXxYyZz
COMPRESSED-DOUBLE WIDTH MODE
0123456789 AaBbCcDdEeFfGgHhIiJjKkLlMmNnOoPpQqRrSsTtUuVvWwXxYyZz
COMPRESSED-DOUBLE WIDTH/STRIKE
0123456789 AaBbCcDdEeFfGgHhIiJjKkLlMmNnOoPpQqRrSsTtUuVvWwXxYyZz
DOUBLE WIDTH MODE
0123456789  AaBbCcDdEeFfGgHhIiJjKkLlMmNn
DOUBLE WIDTH-DOUBLE STRIKE MODE
0123456789  AaBbCcDdEeFfGgHhIiJjKkLlMmNn
DOUBLE WIDTH-EMPHASIZED MODE
0123456789  AaBbCcDdEeFfGgHhIiJjKkLlMmNn
DOUBLE WIDTH/STRIKE-EMPHASIZED MODE
0123456789  AaBbCcDdEeFfGgHhIiJjKkLlMmNn
```

Program.

```
10  REM  EPSON GRAPHTRAX+
20  REM  PRINT MODE SET UTILITY
50  DIM PS$(14),DS$(14)
60  REM   TITLES HERE
70  TEXT : HOME : SPEED= 255:D$ =  CHR$ (4):
      PRINT D$;"NOMON C,I,O": HOME
80  INVERSE : FOR I = 1 TO 4: PRINT  SPC( 40
     ): IF I = 2 THEN  PRINT "  EPSON PRINTE
     R GRAPHTRAX+ SET UTILITY  ";
90  NEXT I: NORMAL : POKE 34,5
100  VTAB 10: PRINT "PLEASE TURN YOUR PRINTE
     R ON NOW."
110  GOSUB 130: GOSUB 140: GOTO 200
120  REM   HIT ANY KEY TO CONTINUE
130  VTAB 20: FLASH : PRINT "HIT
      ANY KEY TO CONTINUE.";: GET A$: PRINT
      CHR$ (1): NORMAL : RETURN
140  REM   PRINT SLOT SET
150 S1 = 1
160  HOME : VTAB 10: PRINT "PRINTER SLOT=";S
     1; CHR$ (7): VTAB 20: PRINT "NEW SLOT (
     #/<CR>)?";: GET NS$: PRINT  CHR$ (1): IF
      ASC (NS$) = 13 THEN 200
170  IF  ASC (NS$) = 27 THEN  TEXT : HOME : END

180  IF  ASC (NS$) < 48 OR  ASC (NS$) > 55 THEN
     160
190 S1 =  VAL (NS$): GOTO 160
200  REM   SET CTRL-STRINGS
210 PS$(1) =  CHR$ (15):DS$(1) = "COMPRESSED
     MODE"
220 PS$(2) =  CHR$ (15) +  CHR$ (27) +  CHR$
     (71):DS$(2) = "COMPRESSED-DOUBLE STRIKE
     MODE"
230 PS$(3) =  CHR$ (27) +  CHR$ (72) +  CHR$
     (18) +  CHR$ (27) +  CHR$ (70):DS$(3) =
     "NORMAL MODE"
240 PS$(4) =  CHR$ (27) +  CHR$ (71):DS$(4) =
     "NORMAL-DOUBLE STRIKE MODE"
250 PS$(5) =  CHR$ (27) +  CHR$ (69):DS$(5) =
     "NORMAL-EMPHASIZED MODE"
260 PS$(6) =  CHR$ (27) +  CHR$ (71) +  CHR$
     (27) +  CHR$ (69):DS$(6) = "NORMAL-EMPH
     ASIZED-DOUBLE STRIKE"
270 PS$(7) =  CHR$ (27) +  CHR$ (14) +  CHR$
     (15):DS$(7) = "COMPRESSED-DOUBLE WIDTH
     MODE"
280 PS$(8) =  CHR$ (27) +  CHR$ (71) +  CHR$
     (14) +  CHR$ (15):DS$(8) = "COMPRESSED-
     DOUBLE WIDTH/STRIKE"
290 PS$(9) =  CHR$ (27) +  CHR$ (14):DS$(9) =
     "DOUBLE WIDTH MODE"
300 PS$(10) =  CHR$ (27) +  CHR$ (71) +  CHR$
     (14):DS$(10) = "DOUBLE WIDTH-DOUBLE STR
     IKE MODE"
310 PS$(11) =  CHR$ (27) +  CHR$ (69) +  CHR$
     (14):DS$(11) = "DOUBLE WIDTH-EMPHASIZED
     MODE"
320 PS$(12) =  CHR$ (27) +  CHR$ (71) +  CHR$
     (27) +  CHR$ (69) +  CHR$ (14):DS$(12) =
     "DOUBLE WIDTH/STRIKE-EMPHASIZED MODE"
330 PS$(13) =  CHR$ (27) + "4":DS$(13) = "SE
     T ITALICS ON"
340 PS$(14) =  CHR$ (27) + "5":DS$(14) = "SE
     T ITALICS OFF"
350  REM   SAMPLE OR JUST SET MODE
360  HOME
370  VTAB 10: PRINT "DO YOU WANT A SAMPLE OF
      ALL THE PRINT": PRINT "STYLES AVAILABL
     E? (Y/N)  ";: GET Y$: PRINT Y$: PRINT  CHR$
     (1): IF Y$ = "Y" THEN 410
380  IF Y$ = "N" THEN 580
390  GOTO 370
400  REM    PRINT SAMPLE
410  PRINT D$;"PR#";S1
420  PRINT  CHR$ (8);"80N"; CHR$ (12)
430  PRINT PS$(11);"  THE    EPSON    MX-SER
     IES    PRINTER"
440  FOR I = 48 TO 57:T$ = T$ +  CHR$ (I): NEXT
     I:T$ = T$ + "  ": FOR I = 65 TO 90:T$ =
     T$ +  CHR$ (I):T$ = T$ +  CHR$ (I + 32)
     : NEXT I
450  PRINT PS$(3)
460  FOR I = 1 TO 2
470  IF I = 1 THEN  PRINT PS$(3);"STANDARD C
     HARACTERS"
480  IF I = 2 THEN  PRINT PS$(3);PS$(13);"IT
     ALIC CHARACTERS"
490  PRINT
500 S$ = T$
510  FOR J = 1 TO 12: IF J > 8 THEN S$ =  LEFT$
     (T$,39)
520  PRINT PS$(3);DS$(J): PRINT PS$(J);S$
530  NEXT J
540  PRINT
550  NEXT I: PRINT PS$(3);PS$(14); CHR$ (12)
     ; CHR$ (12)
560  PRINT D$;"PR#0"
570  REM    SET PRINT STYLE
580  HOME : VTAB 7: FOR I = 1 TO 14: IF I >
     9 THEN  PRINT I;". ";DS$(I)
590  IF I < 10 THEN  PRINT " ";I;". ";DS$(I)

600  NEXT I
610  PRINT I;". EXIT PROGRAM"
620  POKE 34,22
630  ONERR  GOTO 640
640  VTAB 23: INVERSE : INPUT "SELECT PRINT
     STYLE (1-15) : ";P: IF P < 1 OR P > 15 THEN
      PRINT  CHR$ (7): GOTO 640
650  NORMAL
660  IF P = 15 THEN  TEXT : HOME : END
670  IF P < 7 OR P > 12 THEN 760
680  REM   DOUBLE WIDTH LINE PRINT
690  VTAB 7: CALL  - 958: VTAB 8: PRINT "MOD
     ES 7-12 ARE ONE LINE MODES ONLY.": POKE
     34,6: PRINT : PRINT "THE DOUBLE WIDTH C
     HARACTER CONTROL": PRINT "STRING MUST B
     E PRINTED AT THE FRONT OF": PRINT "EACH
      LINE OF WIDE TEXT."
700  PRINT : PRINT "YOU MAY NOW TYPE A 40 CH
     ARACTER (OR": PRINT "LESS) LINE AND IT
     WILL BE PRINTED IN ": PRINT "THE DESIRE
     D MODE.": PRINT : PRINT "PLEASE ADJUST
     YOUR PRINTER PAPER TO THE"
710  PRINT "DESIRED POSITION FOR THE LINE.":
      VTAB 21: PRINT "TYPE IN THE LINE YOU W
     ANT PRINTED:"
720  VTAB 22: INPUT "";L$: IF  LEN (L$) > 40
      THEN  VTAB 22: CALL  - 868: VTAB 22: GOTO
     720
730  PRINT D$;"PR#";S1: PRINT PS$(3);PS$(P);
     L$;PS$(3): PRINT D$;"PR#0": GOTO 580
740  REM   CLEAR PREVIOUS MODE &
750  REM  SEND NEW MODE
760  PRINT D$;"PR#";S1: PRINT PS$(3);PS$(P):
      PRINT D$;"PR#0": VTAB 1: CALL  - 868: VTAB
      1: INVERSE : HTAB ((40 -  LEN (DS$(P)))
      / 2): PRINT DS$(P): GOTO 640
```

©

Apple II Bar Charts

Bernard L. Webb

Reports for school, business, or other purposes can frequently be made more interesting and more understandable by the use of charts and graphs of various types. Such charts can be prepared on the Apple II using a graphics pad or by keyboard control. This fast and convenient program draws bar charts under program control, with only a minimum of effort by the operator.

This program is interactive, with the operator providing on request the necessary information as to the number of bars and so forth. The information requested is quite simple and requires no advance calculations. Error trapping routines have been included to catch most operator errors and prevent premature termination of the program.

Completed charts can be stored as binary files, and other charts can be recalled from disk if desired. The latter function is useful if you want to superimpose a bar chart on a line graph or other illustration previously prepared.

Designing The Chart

The program will draw up to 30 bars in its present form. A larger number of bars can be accommodated by changing the dimension statements at the beginning of the program. However, the limitations of the Apple hi-res screen would cause the bars on even a 30-bar chart to be rather narrow.

In order to provide variety, several types of bar charts can be prepared by the program. The bars can be all positive, as shown in Charts 1 and 4, or they can be both positive and negative, as shown in Charts 2 and 3. The bars can be vertical, as shown in Charts 1 and 2, or horizontal, as shown in Charts 3 and 4. The bars can be contiguous, as shown in Chart 1; or spaces can be inserted between them, as shown in Chart 2. The number of spaces, if any, to be inserted between bars is determined by the operator at runtime.

Generally, it is better to use vertical bars if there are many bars. Horizontal bars are best if the bars need to be long, especially if there are not many of them. Of course esthetics may also be a factor in selecting between them. The minimum value on the bars need not be zero if all bars are positive. If negative bars are included, the chart is centered at zero in order to avoid misleading comparisons.

This program does not provide any means for lettering the charts. The charts shown here were lettered with a VersaWriter digitizer board and the associated software. I sometimes use the *Higher Text Plus* software from Call-A.P.P.L.E. for the lettering, and there are several other software packages that can serve the same purpose.

Printing Options

I print the charts on either my Epson MX-80 dot matrix printer or my NEC Spinwriter letter-quality printer. Chart 1 was printed on the Epson, and Chart 2 was printed on the Spinwriter.

I use the *Grafpak* screen-dump package from Smartware to print charts on the Epson, and the *Spinwriter Graphics* dump program from Computer Station on the NEC. Both have been highly satisfactory, but several other publishers have similar software. Computer Station's software will not work with the serial port of the Mountain Computer CPS MultiFunction Board. The menu says it will work only with the Apple Serial Board, the Apple Communications Card and the California Computer Serial Card. However, I have found that it will also work with the serial port of my SSM AIO card, provided I choose the Apple Communications Card option from the menu.

Figure 1.

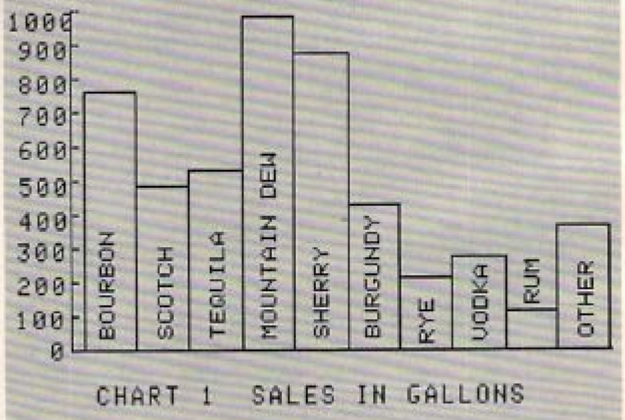


Figure 2.

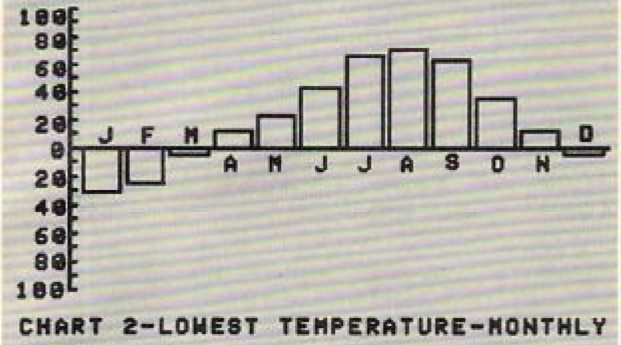


Figure 3.

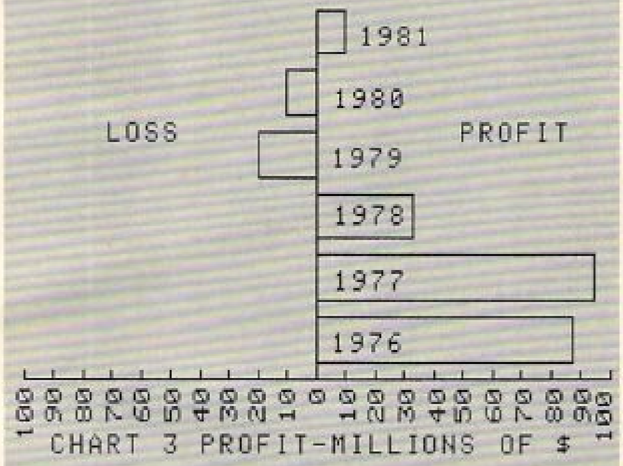
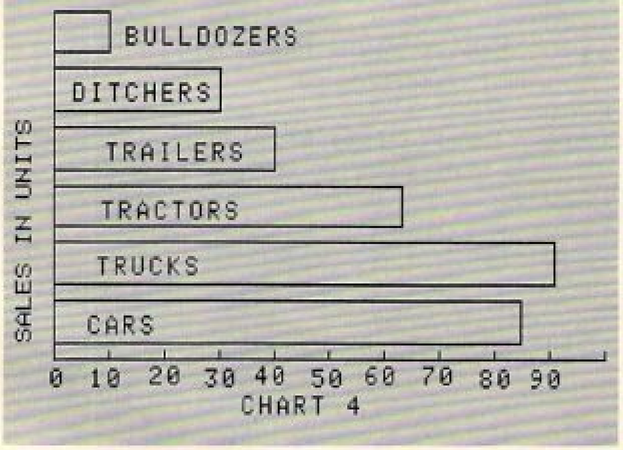


Figure 4.



```
10  DIM YV(30),YP(30),YQ(30)
20  DIM XP(30),XQ(30)
30  HCOLOR= 7
40 D$ =  CHR$ (4):F9 = 0
50  HGR
60 X = 29:Y = 130
70 H = 130:L = 250
80  INPUT "WANT TO RECALL PICTURE
      FROM DISK?(Y/N)   ";Y$
90  IF Y$ = "N" THEN 130
100  IF Y$ <  > "Y" AND Y$ <  > "
     N" THEN 80
110  INPUT "INPUT FILENAME  ";P$
120  PRINT D$;"BLOAD";P$
130  PRINT "DO YOU WANT HORIZONTA
     L OR VERTICAL BARS?(H/V)  "
140  INPUT H$
150  IF H$ <  > "H" AND H$ <  > "
     V" THEN 130
160  INPUT "WILL THERE BE NEGATIV
     E BARS?(Y/N)   ";NB$
170  IF H$ = "V" AND NB$ = "N" THEN
     210
180  IF H$ = "V" AND NB$ = "Y" THEN
     940
190  IF NB$ <  > "N" AND NB$ <  >
     "Y" THEN 160
200  GOTO 1150
210  IF NB$ = "Y" THEN 270
220 XY$ = "Y"
230  HPLOT (X + L),Y
240  HPLOT  TO X,Y
250  HPLOT  TO X,(Y - H)
260  GOTO 310
270  HPLOT (X + L),(Y - (H / 2))
280  HPLOT  TO X,(Y - (H / 2))
290  HPLOT X,Y

300  HPLOT  TO X,(Y - H)
310  INPUT "NUMBER OF BARS ON X A
     XIS? ";N2
320  INPUT "NUMBER OF SPACES BETW
     EEN BARS?  ";SP
330  IF NB$ = "Y" THEN 350
340  INPUT "NUMBER OF DIVISIONS O
     N Y AXIS? ";N1
350  IF NB$ = "Y" THEN Z3 = 2
360  IF NB$ = "Y" THEN Z3 = 2
370  IF NB$ = "N" THEN Z3 = 1
380 Y1 = H / (Z3 * N1)
390 X1 = (L - 6 - (SP * N2)) / N2

400 Y2 = Y:X2 = X
410  IF NB$ = "Y" THEN Z6 = (N1 *
     2)
420  IF NB$ = "N" THEN Z6 = N1
430  IF NB$ = "N" THEN 460
440  HPLOT X,Y
450  HPLOT  TO (X + 3),Y
460  FOR I = 1 TO Z6
470 Y2 = Y2 - Y1
480  IF NB$ = "N" THEN 500
490  IF I = N1 THEN 550
500  IF Y2 < 0 THEN Y2 = 0
510  HPLOT X,Y2
520  HPLOT  TO (X + 3),Y2
530 XP(1) = X + 6
540 XQ(1) = XP(1) + X1
```

```
550  NEXT I
560  FOR I = 2 TO N2
570 XP(I) = XQ(I - 1) + SP
580 XQ(I) = XP(I) + X1
590  IF XQ(I) > 279 THEN XQ(I) =
     279
600 X2 = X2 + X1 + SP
610  NEXT I
620  INPUT "MAXIMUM VALUE SHOWN O
     N Y AXIS? ";Y3
630  IF NB$ = "Y" THEN 660
640  INPUT "MINIMUM VALUE SHOWN O
     N Y AXIS? ";Y4
650  IF Y3 <  = Y4 THEN  GOSUB 16
     30: GOTO 620
660  FOR I = 1 TO N2
670  PRINT "INPUT THE Y-VALUE FOR
      BAR NUMBER  ";I
680  INPUT YV(I)
690  IF YV(I) > Y3 THEN  GOSUB 97
     0: GOTO 670
700  IF Y4 > 0 AND YV(I) < Y4 THEN
      GOSUB 1030: GOTO 670
710  IF Y4 = 0 AND  ABS (YV(I)) >
     Y3 THEN  GOSUB 1090: GOTO 67
     0
720  IF NB$ = "Y" THEN Z7 = H / 2

730  IF NB$ = "N" THEN Z7 = H
740 YQ = ((YV(I) - Y4) / (Y3 - Y4
     )) * Z7
750  IF NB$ = "N" THEN 780
760  HPLOT XP(I),(Y - (H / 2))
770  GOTO 790
780  HPLOT XP(I),Y
790  IF NB$ = "Y" THEN Z8 = H / 2

800  IF NB$ = "N" THEN Z8 = 0
810  HPLOT  TO XP(I),(Y - YQ - Z8
     )
820  HPLOT  TO XQ(I),(Y - YQ - Z8
     )
830  IF NB$ = "N" THEN 860
840  HPLOT  TO XQ(I),(Y - (H / 2)
     )
850  GOTO 870
860  HPLOT  TO XQ(I),Y
870  NEXT I
880  INPUT "WANT TO SAVE PICTURE
     TO DISK?(Y/N)   ";Y$
890  IF Y$ = "N" THEN  END
900  IF Y$ <  > "Y" THEN 880
910  INPUT "INPUT FILENAME  ";X$
920  PRINT D$;"BSAVE ";X$;",A8192
     ,L8192"
930  GOTO 1650
940  INPUT "NUMBER OF DIVISIONS O
     N THE POSITIVE Y AXIS?  ";N1
950 Y4 = 0
960  GOTO 210
970  PRINT "INPUT VALUE EXCEEDS M
     AXIMUM VALUE ON ";XY$;" AXIS
     . WANT TO INPUT NEW VALUE (N
     ) OR TERMINATE (T)?"
980  INPUT T$
990  IF T$ = "N" THEN 1020
1000  IF T$ = "T" THEN  END
1010  GOTO 970
1020  RETURN
1030  PRINT "INPUT VALUE IS LESS
     THAN MINIMUM ";XY$;" VALUE.
     WANT TO INPUT NEW VALUE (N)
     OR TERMINATE (T)?"
1040  INPUT T$
1050  IF T$ = "N" THEN 1080
1060  IF T$ = "T" THEN  END
1070  GOTO 1030
1080  RETURN
1090  PRINT "INPUT NEGATIVE VALUE
      WILL PLOT OFF CHART. WANT T
     O INPUT NEW VALUE (N) OR TEM
     INATE (T)?"
1100  INPUT T$
1110  IF T$ = "N" THEN 1140
1120  IF T$ = "T" THEN  END
1130  GOTO 1090
1140  RETURN
1150  INPUT "HOW MANY BARS ON THE
      Y AXIS?";YB
1160  INPUT "HOW MANY SPACES BETW
     EEN BARS?";NS
1170 XY$ = "Y"
1180  IF NB$ = "Y" THEN 1210
1190  INPUT "HOW MANY DIVISIONS O
     N X AXIS?";XD
1200  GOTO 1230
1210  INPUT "HOW MANY DIVISIONS O
     N POSITIVE X AXIS?";XD
1220 XD = XD * 2
1230  HPLOT (X + L),Y
1240  HPLOT  TO X,Y
1250  IF NB$ = "N" THEN  HPLOT  TO
     X,(Y - H)
1260  IF NB$ = "Y" THEN  HPLOT (X
      + (L / 2)),Y: HPLOT  TO (X +
     (L / 2)),(Y - H)
1270  HPLOT X,Y: HPLOT  TO X,(Y -
     3)
1280 Z = X
1290  FOR I = 1 TO XD
1300 Z = Z + (L / XD)
1310  IF I = XD AND NB$ = "Y" THEN
     1340
1320  IF I = (XD / 2) AND NB$ = "
     Y" THEN 1340
1330  HPLOT Z,Y: HPLOT  TO Z,(Y -
     3)
```

```
1340  NEXT I
1350  HPLOT (X + L),Y: HPLOT  TO
     (X + L),(Y - 3)
1360 ZQ = H - (NS * YB)
1370  IF NB$ = "Y" THEN 1420
1380  INPUT "INPUT MAXIMUM VALUE
     ON X AXIS";MX
1390  INPUT "INPUT MINIMUM VALUE
     ON X AXIS";LX
1400  IF MX <  = LX THEN  GOSUB 1
     630: GOTO 1380
1410  GOTO 1430
1420  INPUT "INPUT MAXIMUM VALUE
     ON POSITIVE X AXIS";MX
1430 ZZ = ZQ / YB
1440 YP(1) = Y - 6
1450 YQ(1) = Y - 6 - ZZ
1460  FOR I = 2 TO YB
1470 YP(I) = YQ(I - 1) - NS
1480 YQ(I) = YP(I) - ZZ
1490  IF YQ(I) < 0 THEN YQ(I) = 0

1500  NEXT I
1510  IF NB$ = "Y" THEN ZV = X +
     (L / 2):ZY = L / 2
1520  IF NB$ = "N" THEN ZV = X:ZY
      = L
1530  FOR I = 1 TO YB
1540  PRINT "INPUT X VALUE FOR BA
     R NO. ";I
1550  INPUT BI
1560  IF BI > MX THEN  GOSUB 970:
      GOTO 1540
1570  IF NB$ = "N" AND BI < LX THEN
      GOSUB 1030: GOTO 1540
1580  IF NB$ = "Y" AND  ABS (BI) >
     MX THEN  GOSUB 1090: GOTO 15
     40
1590 ZW = ZV + ((BI - LX) / (MX -
     LX)) * ZY
1600  HPLOT ZV,YP(I): HPLOT  TO Z
     W,YP(I): HPLOT  TO ZW,YQ(I):
      HPLOT  TO ZV,YQ(I)
1610  NEXT I
1620  GOTO 880
1630  PRINT "MINIMUM VALUE IS EQU
     AL TO OR GREATER THAN MAXIMU
     M. PLEASE TRY AGAIN."
1640  RETURN
1650  END
```
©

VIC – ATARI – APPLE – SINCLAIR

JOIN THE USER'S NETWORK

GET: * DISCOUNTS & $20 IN COUPONS
* TECH HELP & PROGRAMS FROM MEMBERS
* TO COMMUNICATE BY MODEM
* USER'S-NETWORK MEMBERS DIRECTORY
* NEWSLETTER & CONTENTS
* YOUR ARTICLE OR PROGRAM PUBLISHED

SEND: NAME, ADDRESS, PHONE #, LIST OF HARDWARE AND MAKE OF COMPUTER PLUS $5.00 TO:
THE USER'S NETWORK, 3 LAUREL ST, N PROVIDENCE RI 02911 PHONE # (401)353-6174

SECUR-A-TEXT™

THE SECURITY
PROTECTION
FOR YOUR FILE DISKETTES

SECUR-A-TEXT is a high level security program designed to protect any text file from being loaded into your computer. A 32-digit number sequence must be entered into your computer before your text file can be viewed on your screen.
Available for Apple II/II Plus® (3.3 DOS)

Retail $49.95

Check your local dealer or send check or M.O. with $2.00 postage and handling. California residents add 6½% sales tax.

T & F SOFTWARE COMPANY
A Talcove & Familian Company
10902 Riverside Drive • North Hollywood, CA 91602
(213) 501-5845

Copyright © 1982 Megagem • For sale in U.S. only
Apple is a registered trademark of Apple Computers Inc.

Keith Falkner

Load Commodore BASIC Tapes Into Apple II

This month's column includes an extraordinary program which will let Apple users load PET/CBM programs directly off tape into the Apple. *This opens up a world of new software. The column is also of interest to Commodore users because it includes numerous explanations of the differences between the two computers' BASICs and software.*

Between microcomputers, compatibility is rare. **COMPUTE!** sets a good example by publishing BASIC listings of programs for several different machines. But why not go further and teach one computer to load programs which already exist in a different computer? This approach instantly gives access to some of the other computer's software base. Two years ago I wrote a program to load PET BASIC programs into the Apple, and made it available to readers of **COMPUTE!**. Now there are new computers from Commodore, so it's time to replace that program with one which can read any Commodore BASIC tape. Commodore seems to be retaining the current tape format, so this program may stay current.

Briefly, here is how to use it. The Commodore BASIC Tape Loader appears to be an Applesoft program, and can be loaded, saved, and run as an Applesoft program, although it is written entirely in Assembly Language. When run, it prints simple instructions and messages in English. It loads a Commodore BASIC program from tape into memory, and converts most of it into Applesoft. Then it ends, and you can do what you wish with the loaded program. You will likely choose to save it to disk, with the hope of completing the conversion at your leisure. In any case, the Commodore Loader will not again be needed for that program. The Commodore Loader can be rerun without being reloaded.

Tape Only

The Commodore Loader handles tape input only, because Commodore disk formats are very complex, and some of them are beyond the ability of an Apple Disk][to read. If you have disks of Commodore programs to convert, use a disk-to-tape program on the Commodore computer to copy many programs to one cassette; tape positioning is not critical.

You can even use the Commodore Loader if there isn't a tape drive or a cassette in sight. Just wire the cassette output line of the Commodore to the cassette input port of the Apple, ground the "sense" line to indicate a recorder is present and operating; then run the Commodore Loader, and issue a SAVE instruction on the Commodore machine. Honest, it works! And if you have a Franklin Ace instead of an Apple, the Ace lacks tape hardware and support for tape in the ROMs. You can use pin four of your Ace game I/O socket, after changing all references in the program from $C060 (cassette input) to $C063 (switch three). Some voltage amplification may be needed, but the lack of tape I/O routines in the ROMs will not matter.

What Can Go Wrong?

Three things can go wrong, all self-explanatory. The program prints a sentence announcing the error, then quits. "Tape is unreadable" means that 128 or more bytes could not be read correctly from tape. Each incorrect byte provoked a click from the speaker, so you can hear this message coming before it finally appears. "End-of-tape was found" means that the computer which wrote the tape put a "Type-5" label here to mark the end of the recorded part of the tape. You are free to rerun and try to load programs beyond that label if you think any are there. "The program is too big" means that the incoming Commodore program will not fit between location 4096 ($1000) and the current HIMEM. Perhaps you can adjust HIMEM or MAXFILES if this occurs, and try again.

The Commodore Loader does not quit if a few (dozen) tape errors occur, because, first, there is no program to recover a tape it can't read, and, second, any bytes read incorrectly from tape are stored as the Applesoft token for ERROR (BEEP), and will actually cause the beep when you LIST the loaded program. If you find what seems like too many of these, consider adjusting the tape volume; it should be high enough to irritate, but not enough to hurt. In practice, even a solitary error of this type is rare, because there are two copies of the program on tape, and the Commodore Loader uses the second copy, if necessary, to correct bytes read erroneously from the first.

Incompatibilities

There are some fundamental differences between Apple and Commodore computers, but fortunately few programs exploit these differences. Adventure programs and scientific calculations need almost no changes. Business programs will need their disk and printer I/O routines largely rewritten. In general, a few changes will be needed for screen formatting, but these are obvious and will soon become routine. Here are specific areas where differences between the computers will necessitate changes in the programs.

1. No equivalent verb in the Apple.

OPEN, CLOSE, VERIFY, and CMD are commands in Commodore BASIC, and are translated to STOP. Machines with Version 4 BASIC have many more I/O commands such as DOPEN, COLLECT, and CATALOG. These are all translated to "UNDEF'D FUNCTION", and you will need to decipher the programmer's intent to program the equivalent for the Apple. Refer to Commodore's fine manuals for descriptions of these commands.

2. Specific device reference.

Programs containing OPEN and CLOSE will contain PRINT# and INPUT# commands, which are simply translated to PR# and IN# respectively, and will require substantial rework. The devices, by number, are conventionally these:

#0	Keyboard
#1	Cassette Recorder
#2	Serial I/O Port
#3	Screen
#4	Printer
#8	Disk Drive

The numbers above are hardware addresses, and are the second numeric operand in the OPEN statement (translated to STOP), so STOP 6,4,128 addressed the printer (address 4) and defined file number 6 (the first operand of OPEN).

3. Reference to actual memory locations.

PEEK, POKE, CALL (SYS in Commodore BASIC), WAIT, and USR refer to specific locations in memory, and you will need more help than I can offer here.

4. Keys to move the cursor.

Commodore computers have keys to control the position, color, or action of the cursor. These are translated as follows:

Two functional equivalents:

CURSOR-LEFT becomes BACKSPACE, and appears as GR in the program.

CURSOR-DOWN becomes LINE-FEED, and appears as PR# in the program.

(Odd as they appear, these actually move the cursor exactly as stated.)

One destructive approximation:

CURSOR-RIGHT becomes SPACE, which obliterates what it should space past. This appears as COLOR= in the program.

Seven nonfunctional comments:
These keys are translated into Applesoft tokens selected to indicate what key was pressed by the programmer: 'RVS' -> INVERSE, 'OFF' -> NORMAL, 'HOME' -> HOME, 'CLR' -> CLEAR, 'CRSR UP' -> VLIN, 'DEL' -> DEL, and 'INST' -> IN#.

When the program is listed, these are visible, looking like genuine verbs, and it looks as if the name of the key will be printed. For example,

```
100 PRINT " CLEAR "
110 INPUT " INVERSE INSTRUCTIONS NORMAL
    ";Z$
```

In fact, line 100 will neither clear the screen nor print "CLEAR". It will merely print an equal sign (=). Line 110 will print "INSTRUCTIONS", and no trace will be seen of the INVERSE and NORMAL commands shown in the listing. This behavior can be perplexing, because, usually with Applesoft, what you see is what you get. The purpose of these translations is to disclose the programmer's intent, so when you list the program and discover:

```
300 INPUT " HOME PLAY INVERSE AGAIN
    NORMAL " ; X$
```

substitute the equivalent Apple code, which in this case is:

```
300 VTAB 1: PRINT "PLAY ";: INVERSE: PRINT
"AGAIN";: NORMAL: INPUT "? ";X$
```

For all those keys except INST and DEL, the equivalent in Applesoft is easy to devise, but simulating the Commodore computer's INSERT and DELETE keys is extremely difficult, and Apple's convoluted screen addressing makes this task truly difficult. Fortunately, very few Commodore BASIC programs print INSERT or DELETE characters.

5. Printing of numbers is slightly different.

```
290 X=4 : Y= -6
300 PRINT "X IS" ; X ; "Y IS" ; Y ; "."
```

```
Commodore BASIC:     X IS 4 Y IS-6 .
The Apple prints:    X IS4Y IS-6.
```

Commodore computers print a blank before positive numbers, and a CURSOR-RIGHT after all numbers; the Apple does neither. By the way, all four semicolons (;) in line number 300 above are optional in Apple and Commodore computers.

6. A side effect of TAB.

Commodore computers TAB over data already on the screen; the Apple wipes it out, so substitute an HTAB verb for a TAB phrase if the problem occurs.

```
CBM:   40 PRINT TAB(12) "XYZ"
Apple: 40 HTAB 12 : PRINT "XYZ"
```

7. Computations in Boolean arithmetic.
In the following lines,

```
400 X=11 : Y=6 : Z=X > Y
410 PRINT Z : IF Z THEN 500
```

Z is -1 in Commodore BASIC, so line 410 will print this result and go to 500. The Apple will set Z to +1, and print this different result, then go to 500. In the above example, the difference may not be crucial, but it often can be. Commodore BASIC does bit-by-bit evaluation of the operators OR and AND, so in

```
700 X=11 : Y=6 : Z=X AND Y
```

Commodore BASIC sets Z=2 because the bit pattern of 11 is 0000 1011 and the bit pattern of 6 is 0000 0110, and these two patterns, ANDed, give 0000 0010, arithmetically 2. Apple, on the other hand, merely sees that neither X nor Y is FALSE (zero), calls the result TRUE, and sets Z equal to 1. This can be a very subtle pitfall.

8. Random numbers.

RND (0) gives a genuine random number each time in Commodore BASIC, but in an Apple it repeats the previous random number. Simply replace the 0 with a 1.

9. The GET command.

Commodore BASIC's GET does not wait for a key to be struck, so the sequence

```
333 GET P$ : IF P$ = "" THEN 333
```

is the customary way to wait for a key to be typed. This same sequence is completely appropriate in the Apple, because if the key struck is CTRL-@, then P$ will be the null string. Ignorance of this is an obscure bug in some Applesoft programs. When the program is testing for a key, but not waiting when no key has been struck, a different approach is needed. For example,

```
CBM:   60 GET A$ : IF A$ = "" THEN 100
Apple: 60 ON PEEK (-16384) < 128 GOTO 100 : GET
          A$ : IF A$="" THEN 60
```

10. Graphics characters and lowercase.

Commodore computers can display lowercase letters and many symbols which the Apple cannot, and there are two display modes: Text and Graphic. $C1 is "a" (old machine), "A" (any other), or the symbol for the Spade suit (any PET). The Loader looks for $CF, probably a lowercase "o," in the program. If $CF is found, all letters are translated to uppercase; if not, graphic symbols are translated into a similar character the Apple can produce.

11. Direct screen addressing.

In many computers, the video screen occupies a part of memory, and a POKE to a storage location in the screen memory will produce a character on the screen. The relationship between memory location and screen position in Commodore computers is straightforward, but it is complex in the Apple, hence not often used. Nevertheless, it is worth mastering, because there are hordes of programs which use this technique, and a lot of them are attractive games.

The PET has 25 lines of 40 columns, and the memory location of each byte of the screen can be computed thus (the expression is not written in BASIC):

```
LOCATION=32768+40 * LINE+COLUMN
```

where the upper left corner is LINE 0, COLUMN 0.

The 80-column SuperPET and CBM computers can use similar addressing to POKE data onto the screen; just change the 40 to 80. The VIC-20 has 23 lines of 22 columns each, but the address of the screen depends on what expansion memory units are installed. The Commodore 64 has again 25 lines of 40 columns, but this machine can decide where in memory the screen is. The default address is 1024 (same as Apple's), so change 32768 to 1024 and the formula shown above applies to the 64 in its default state.

The Apple has 24 lines of 40 columns, and the memory location of each byte can be calculated by:

```
LOCATION=XL% ( LINE ) +COLUMN
```

where the array XL% has been initialized thus:

```
1000 DIM XL% (23) : FOR I=0 TO 7 :
     XL% (I)      =1024+128 * I :
     XL% (I+  8)  =1064+128 * I :
     XL% (I+16)   =1104+128 * I : NEXT
```

As before, the upper left corner is LINE 0, COLUMN 0. In applying this tactic, take care not to let COLUMN exceed 39, or you will cause destruction of some important values in memory. For example, a POKE to valid LINE 23, and invalid COLUMN 49, will likely cause loss of your BASIC program.

12. Numeric keypad.

Programs which use screen-POKEs to move pieces of a game around the screen use the keys 1-9 to indicate the direction of motion. This is satisfactory on the PET, because these keys form a square, with 7, 8, 9 above 4, 5, 6 above 1, 2, 3, and it is natural that, if the 5-key means "stop," the 8-key means "up," and the 3-key means "down and right." Apple's numeric keys do not form a square, so some substitute must be devised. None is immediately obvious, but perhaps the parallelogram formed by R, T, Y above F, G, H above V, B, N would serve, since the BELL on the G-key can be easily remembered as being a home position. Often the game can be improved by substituting use of the game paddles or joystick. This is where you can improve on the original program.

13. Sound.

Stock Commodore computers are silent or have a rudimentary speaker. The various types of computers have their own ways of making noise, typically with a trio of POKE commands to 59464, 59466, and 59467. The VIC-20 uses similar methods, but the addresses are different. The 64 has a powerful sound chip which you won't try to simulate. Except for the VIC-20, the various POKE commands associated with production of sound address the Apple's Read-Only-Memory, which is unaffected by POKE.

14. Realtime clock.

Commodore computers have a realtime clock, which programs can read and set. TI increases by 60 every second, and can be read but not written; string TI$ is six numeric characters, in the format HHMMSS, and can be read and written.

```
100 INPUT "WHAT TIME IS IT" ; U$ : TI$ = U$
```

When this is executed, TI is set consistent with the value put in TI$.

```
400 PRINT "THE TIME IS " ; TI$
```

TI$ is computed from the instantaneous value of TI, and formatted as six digits. If you try running a PET program in your APPLE, and it just stalls, doing nothing at all, press CTRL-C to stop it, and you may find lines like

```
700 X = TI + 60
710 IF TI < X THEN 710
```

Line 710 is merely waiting for a second to elapse, and in the timeless Apple, it never will. Substitute a FOR/NEXT loop of the appropriate duration.

15. PI.

A single key on the PET provides the number PI, 3.14159265. No such facility exists in the Apple, so PI is translated into a character which prints as "UNDEF'D FUNCTION". This causes "?SYNTAX ERROR", and is simple to correct.

16. The INPUT statement has subtle differences.

Commodore BASIC supplies a question mark after the prompt; and, for neatness, you should supply one. In some Commodore computers, if an INPUT statement is issued, and the user accidentally or deliberately gives a null response by just pressing RETURN, execution of the program ceases at once. This is such a nuisance that programs with any elegance guard against it in a variety of ways, for example:

```
50 INPUT "WHAT NOW > > + < < < ";X$
```

where ">" and "<" stand for keys to move the cursor right and left. You can tidy this up as you supply the question mark for the prompt.

Does It Work?

The above list of incompatibilities is not exhaustive, but it is lengthy. Thus, you may wonder if the Commodore Loader is of much use. In practice, only a very small proportion of the statements in a program require conversion, and the conversion is usually noncritical.

The Commodore BASIC Tape Loader is a useful tool, and permanently gives you access to the expanding world of public-domain Commodore Software. In short, it's worth keying into your Apple.

Operating Instructions

1. Before loading the Commodore BASIC Tape Loader, insure that Applesoft's memory pointers are normal, by issuing "FP" or by booting the System Master. This is necessary because the machine language program cannot operate at any address except the one it was assembled into.
2. Load and run the Commodore BASIC Tape Loader. It asks you to either ready a cassette and press PLAY or to press ESC to exit. Take your choice.
3. Soon after the tape starts moving, the message "OK. SEARCHING ..." is displayed. Clicks from the speaker indicate unreadable data on the tape, and that is a normal condition before or between programs on that tape.
4. If a Data File (not a program) is detected, the program will display the File's name, bypass the data file, and keep searching for a program.
5. When a program is detected, its name is displayed, and the program is loaded into memory, and simultaneously displayed in a little window on the screen. Some of it will be legible. This can take several minutes.
6. If any bytes are unreadable, the Commodore Loader will need to process the second image of the program on tape, so your wait will be doubled.

7. When the Apple beeps, stop the tape. You now have a Commodore BASIC program in your Apple. Decide what to do with it.
8. To load another program, just type "&" (ampersand) and press RETURN. Unless something drastic has occurred, you needn't reload the Loader.

Making It Work

You can key in all the hex digits using the machine language monitor, or better yet, the MINI-ASSM described in this column last December.

So, now you've gotten it correctly assembled and saved; here is how to package it as an Applesoft program so that it can be used with the usual RUN and LOAD and SAVE commands:

```
Get the object program into memory.
CALL -151
67:01 08 FF 0F FF 0F FF 0F
AF:FF 0F N E003G
SAVE COMMODORE LOADER
```

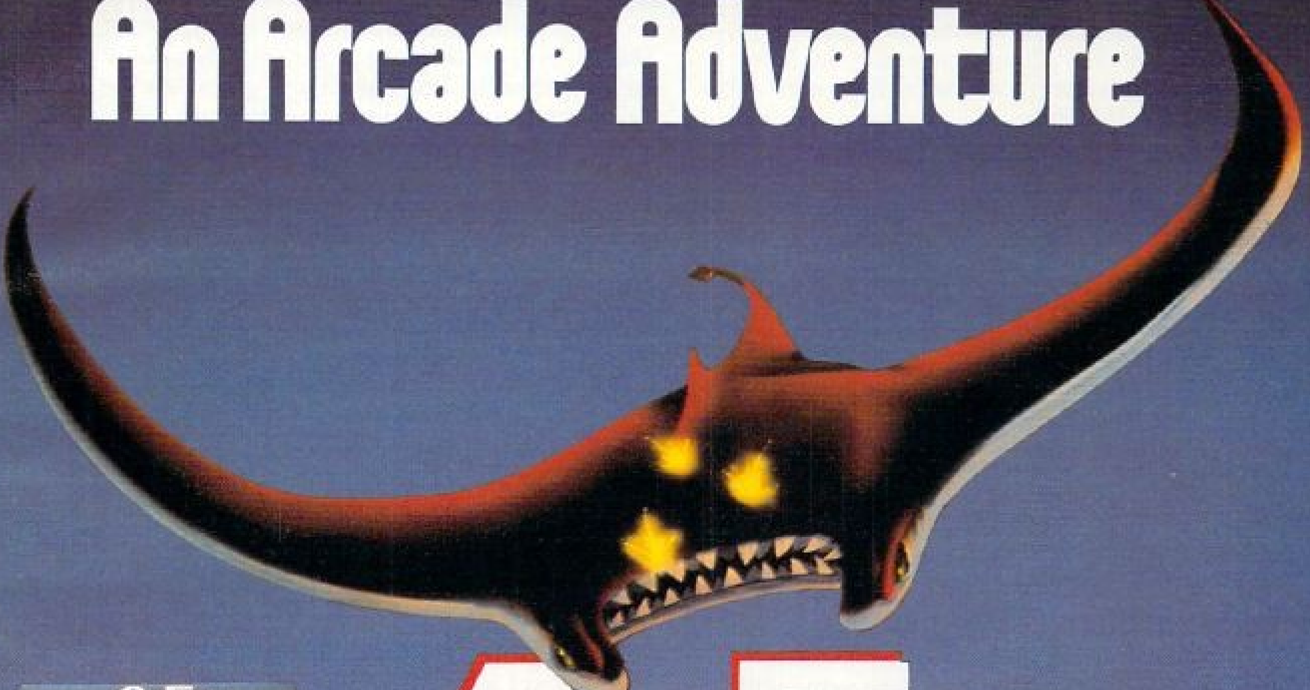
A LIST command now should show only one line: 6502 CALL 2061.

PET To Apple Loader.

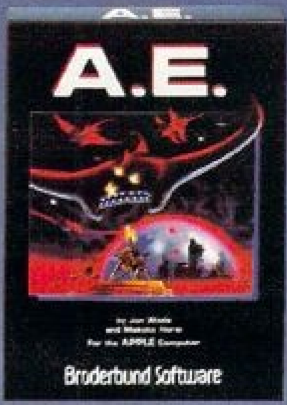
```
0800- 00 0B 08 66 19 8C 32 30  09D0- 08 86 AF A4 09 84 B0 E8  0BA0- 4C 4F 41 44 45 52 2E 20
0808- 36 31 00 00 00 20 2F FB  09D8- D0 01 C8 E8 D0 01 C8 86  0BA8- 20 20 20 20 20 56 20 33
0810- 20 58 FC 20 84 FE 78 A9  09E0- 69 84 6A 86 6B 84 6C 86  0BB0- 2E 30 20 28 43 29 20 4B
0818- 4C 8D F5 03 A9 0D 8D F6  09E8- 6D 84 6E A5 67 85 06 A5  0BB8- 45 49 54 48 20 46 41 4C
0820- 03 A9 08 8D F7 03 A9 00  09F0- 68 85 07 A0 01 B1 06 F0  0BC0- 4B 4E 45 52 2C 20 54 4F
0828- 8D 00 10 A9 01 85 67 A9  09F8- 1F A0 04 B1 06 F0 03 C8  0BC8- 52 4F 4E 54 4F 20 43 41
0830- 10 85 68 20 4B D6 A9 25  0A00- D0 F9 98 A0 00 38 65 06  0BD0- 4E 41 44 41 2C 20 31 39
0838- 85 1B A9 19 85 1C A0 00  0A08- 91 06 AA A9 00 65 07 C8  0BD8- 38 33 2E 0D 0D 0D 20 52
0840- 20 AE 0A 8C 10 C0 AD 60  0A10- 91 06 86 06 85 07 D0 DB  0BE0- 45 41 44 59 20 41 20 43
0848- C0 85 2F A0 00 AD 60 C0  0A18- A5 67 85 06 A5 68 85 07  0BE8- 41 53 53 45 54 54 45 20
0850- 45 2F 10 16 45 2F 85 2F  0A20- A0 00 B1 06 AA C8 B1 06  0BF0- 41 4E 44 20 50 52 45 53
0858- A9 AE 20 F6 FD C8 C0 50  0A28- F0 78 88 84 A6 86 08 85  0BF8- 53 20 27 50 4C 41 59 27
0860- 90 08 A9 8D 20 F6 FD 4C  0A30- 09 18 A5 06 69 04 85 06  0C00- 20 4F 52 0D 0D 20 50 52
0868- 89 08 AD 00 C0 C9 9B D0  0A38- A5 07 69 00 85 07 B1 06  0C08- 45 53 53 20 27 45 53 43
0870- DC A9 04 20 5B FB 20 42  0A40- C9 22 F0 43 24 A6 30 16  0C10- 27 20 54 4F 20 45 58 49
0878- FC A0 C8 20 AE 0A A9 01  0A48- C9 FC F0 41 C9 CB B0 08  0C18- 54 2E 0D 0D 00 4F 4B 2E
0880- 85 67 A9 08 85 68 4C A7  0A50- AA 10 3A BD A0 0C D0 02  0C20- 20 53 45 41 52 43 48 49
0888- 0A A9 04 20 5B FB 20 42  0A58- A9 FB 91 06 D0 2F A2 18  0C28- 4E 47 20 2E 2E 2E 0D 0D
0890- FC A0 92 20 AE 0A A9 00  0A60- CA DD C7 0C F0 19 CA D0  0C30- 00 46 4F 55 4E 44 20 44
0898- 85 1D 20 BC 0A A0 00 20  0A68- F7 A6 A8 D0 0E C9 E0 B0  0C38- 41 54 41 20 46 49 4C 45
08A0- 15 0B 99 30 02 C8 C0 16  0A70- 0A C9 A0 90 06 E9 A0 AA  0C40- 3A 20 00 4C 4F 41 44 49
08A8- 90 F5 A5 A5 F0 0D A5 1D  0A78- BD E0 0C 29 7F 10 03 BD  0C48- 4E 47 3A 20 00 4F 4B 21
08B0- 30 04 A9 80 30 E2 A9 02  0A80- C8 0C 91 06 4C 8D 0A A5  0C50- 0D 0D 00 54 4F 20 52 45
08B8- 4C 61 0B 85 1D AD 30 02  0A88- A6 49 80 85 A6 E6 06 D0  0C58- 53 54 41 52 54 2C 20 50
08C0- C9 01 F0 32 C9 03 F0 2E  0A90- 02 E6 07 A5 06 C5 08 D0  0C60- 52 45 53 53 20 27 26 27
08C8- C9 04 F0 06 C9 05 F0 21  0A98- A5 A5 07 C5 09 D0 9F 4C  0C68- 20 41 4E 44 20 27 52 45
08D0- D0 C4 A0 A6 20 AE 0A A0  0AA0- 20 0A A0 C2 20 AE 0A 58  0C70- 54 55 52 4E 27 2E 0D 0D
08D8- 00 B9 35 02 09 80 20 F6  0AA8- 8D 10 C0 4C 03 E0 B9 BB  0C78- 00 45 52 52 4F 52 3A 20
08E0- FD C8 C0 10 90 F3 A9 8D  0AB0- 0B F0 08 09 80 20 F6 FD  0C80- 07 00 54 41 50 45 20 49
08E8- 20 F6 FD 20 F6 FD 4C 96  0AB8- C8 D0 F3 60 A9 00 85 A5  0C88- 53 20 55 4E 52 45 41 44
08F0- 08 A9 17 4C 61 0B A9 04  0AC0- 8D 10 C0 A0 0D 20 51 0B  0C90- 41 42 4C 45 2E 0D 00 45
08F8- 20 5B FB 20 42 FC A0 B8  0AC8- E4 1B B0 10 88 10 F6 AD  0C98- 4E 44 2D 4F 46 2D 54 41
0900- 20 AE 0A A0 00 B9 35 02  0AD0- 56 0B 8D 5C 0B 49 20 8D  0CA0- 50 45 20 57 41 53 20 46
0908- 09 80 20 F6 FD C8 C0 10  0AD8- 56 0B D0 E7 20 15 0B C9  0CA8- 4F 55 4E 44 2E 0D 00 54
0910- 90 F3 A9 8D 20 F6 FD 20  0AE0- FC D0 0A AD 00 C0 C9 9B  0CB0- 48 45 20 50 52 4F 47 52
0918- F6 FD 20 F6 FD 20 F6 FD  0AE8- D0 03 4C 71 08 45 1D C9  0CB8- 41 4D 20 49 53 20 54 4F
0920- A9 01 85 06 A9 10 85 07  0AF0- 84 F0 09 24 A5 10 E5 A9  0CC0- 4F 20 42 49 47 2E 0D 00
0928- 38 AD 33 02 ED 31 02 AA  0AF8- 02 4C 61 0B 85 1E A9 00  0CC8- 11 8A 12 9E 13 97 14 B5
0930- AD 34 02 ED 32 02 A8 18  0B00- 85 A5 C6 1E A9 80 C5 1E  0CD0- 1D A0 91 8F 92 9D 93 BD
0938- 8A 69 01 85 08 AA 98 69  0B08- F0 46 20 15 0B 45 1D C5  0CD8- 94 8B 9D 88 FC FC A2 27
0940- 10 85 09 C5 74 90 0F D0  0B10- 1E F0 EF D0 C7 20 51 0B  0CE0- 20 5B 3D 2D 5F 28 2A 29
0948- 04 E4 73 90 05 A9 2F 4C  0B18- E4 1B 90 F9 20 51 0B A9  0CE8- 3D 2F 5D 21 2E 2E 2E 3D
0950- 61 0B A9 00 85 1D A9 00  0B20- 80 85 19 85 1A 20 51 0B  0CF0- 2E 2D 2D 21 5B 5B 5D 3D
0958- 85 A8 A8 91 06 85 A7 20  0B28- 90 02 E6 1A 66 18 20 51  0CF8- 3D 3D 5D 2E 2E 2E 2E 3F
0960- BC 0A 20 15 0B D0 0B A5  0B30- 0B 66 19 90 F0 20 51 0B  0D00- 2D 53 21 2D 2D 2D 2D 28
0968- 08 D0 02 C6 09 C6 08 20  0B38- 90 02 E6 1A A5 19 49 FF  0D08- 29 2E 2E 2E 5B 5C 2F 5B
0970- 15 0B 24 1D 10 04 C9 FC  0B40- C5 1B D0 04 66 1A B0 07  0D10- 5D 2A 5F 48 28 2E 5B 28
0978- F0 04 A0 00 91 06 48 A5  0B48- E6 A5 A9 FC 8D 30 C0 AA  0D18- 43 29 44 2B 5B 21 23 5C
0980- 06 29 1F AA 68 48 09 80  0B50- 60 A2 00 2C 60 C0 10 FB  0D20- 80 81 82 83 88 84 86 87
0988- 9D 04 07 E6 A7 A9 05 C5  0B58- E8 2C 60 C0 30 FA E4 1C  0D28- AA AB AC AD AE B0 B1 B2
0990- A7 90 0D 68 AA D0 02 85  0B60- 60 48 A9 08 20 5B FB 20  0D30- B3 B4 B5 B6 B7 B3 B8 B9
0998- A7 C9 CF D0 04 85 A8 48  0B68- 42 FC A0 00 B9 79 0C F0  0D38- BA BA BB BC BD B3 BC B3
09A0- 68 A5 07 C5 09 D0 06 A5  0B70- 08 09 80 20 F6 FD C8 D0  0D40- B3 BE BF C0 C1 C2 C3 C4
09A8- 06 C5 08 F0 08 E6 06 D0  0B78- F3 68 A8 B9 80 0C F0 08  0D48- C6 C7 C8 C9 CA CB CC CD
09B0- BE E6 07 D0 BA A5 A5 F0  0B80- 09 80 20 F6 FD C8 D0 F3  0D50- CE CF D0 D1 D2 D3 D4 D5
09B8- 0B A5 1D 30 07 A9 80 85  0B88- 4C A7 0A 43 4F 4D 4D 4F  0D58- D6 D9 DA DB DC DD DE DF
09C0- 1D 4C 56 09 20 E2 FB A9  0B90- 44 4F 52 45 20 42 41 53  0D60- E0 E1 E2 E3 E4 E5 E6 E7
09C8- 01 85 67 A9 10 85 68 A6  0B98- 49 43 20 54 41 50 45 20  0D68- E8 E9 EA
```

©

Brøderbund Presents An Arcade Adventure

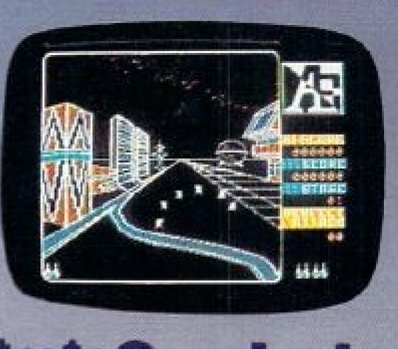
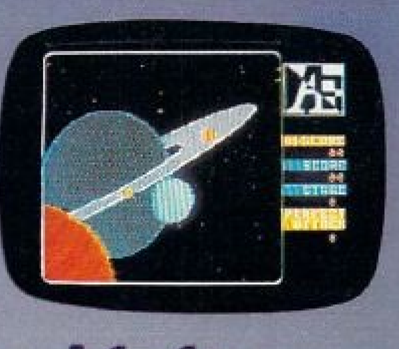


A.E.



A.E.'s, produced by an industrial giant to control pollution on Earth have slipped quality control. They attack relentlessly in waves from the sky. Your mission: to drive them farther and farther into space. With each successful defense, you are transported to another, then another more distant 3-D environment. With these progressively difficult scenes, A.E. delivers the ultimate challenge.

Never has a computer game required such precision, such timing. You'll be hooked from the very start. A.E. (it means sting ray in Japanese) provides such enduring satisfaction on every level that it will become your personal standard of excellence in computer gaming.

Now available for Apple II, II+, IIe and Atari 400/800†

†Apple and Atari are registered trademarks of Apple Computer, Inc., and Atari, Inc. respectively.

Brøderbund Software

Broderbund Software, Inc. 1938 Fourth Street San Rafael, CA 94901 (415) 456-6424

AMERICA'S FAVORITE COMPUTER GAME IS NOW A CARTRIDGE, TOO!

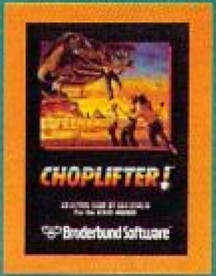
FOR THE ATARI 400/800*

CHOPLIFTER!™

Brilliant animation, dazzling graphics and world-class arcade action have made Choplifter the favorite of tens of thousands of Apple II and Atari 400/800 owners. Previously released only on disk, Choplifter is now available in a convenient plug-in ROM cartridge.

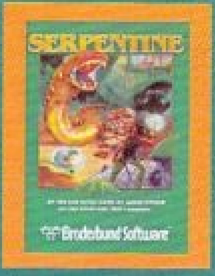
Now you too can unleash the hero within you as you pilot your rescue chopper behind enemy lines, saving your comrades from enemy fire.

Choplifter's detailed, lifelike 3-D graphics will give you a sense of realism unmatched by any other game available today.



SO IS AMERICA'S MOST ADDICTING...

SERPENTINE.™



Yet another Brøderbund hit, Serpentine thrusts you into a terrifying age when mighty serpents ruled the earth! Serpentine will hold your interest through hundreds of plays . . . challenging you at every level.

Join the legion of Choplifter heroes and brave Serpentine warriors and discover a whole new world of arcade action.

Brøderbund products are available at your retailer or by writing to:

Brøderbund Software™

1938 Fourth Street, San Rafael, CA 94901, (415) 456-6424

*Atari 400/800 and Apple II are registered trademarks of Atari, Inc. and Apple Computer, Inc., respectively.

TCON: The Apple Writer Processes Programs

Michael Ginsberg

Would you like to have the power to: change all or some variables in an Apple program; look at two different parts of a program at the same time; find all occurrences of a word or phrase; move one or more lines of a program around at will; have named GOSUB targets; and have other powerful programming tools at your fingertips? You've already got it. Here's how to get more out of the Apple Writer than you may have thought possible.

The Apple Writer, the word processor which comes with every Apple II, can be used in two ways to aid your programming. First, you can use the features of Apple Writer to modify existing programs. Second, you can write your new programs directly using the Apple Writer. If you write programs using the Apple Writer, the only difference is that you use the control-K to keep the characters in uppercase.

A knowledge of text files and BASIC files is necessary to understand how this process works. A short program is included here for files that are currently BASIC programs. This short program uses the EXEC feature of the Apple to create a routine that converts the BASIC program to text so that the Apple Writer can read it.

The TCON program appends three lines to the beginning of your program. The line numbers are 0, 1, and 2. If you already have lines in your program that use those numbers, you must increase these line numbers to 3 or above. First, type in and run EXEC TCON; it will create the TCON program which will convert BASIC to text. Load in the BASIC program and type in EXEC TCON; the disk will start spinning, and your program will be converted. When the program has been converted, you can boot your Apple Writer and use all of the features to help you debug your program. After it is booted, you should hit control-K so it will be in alpha lock.

Some of the features of TCON are: search, replace, scrolling, deleting and retrieving, split screen, and word and phrase counter. Some experimenting with Apple Writer is necessary to learn how it works. After you have finished debugging your program, all you need to do is save the file.

The next step involves converting your file to a BASIC program. This sounds hard but is actually quite simple. After DOS is booted, you need to type NEW; then type EXEC followed by the file name. That's it. Two minutes later, after you've seen many]'s, your file will be magically converted to a working BASIC program. Now you should save the BASIC program and, if you are through making changes, you can delete the text file. Apple Writer can be extraordinarily versatile as a programming aid.

```
10 Q$ =  CHR$ (34):D$ =  CHR$ (4)
20  PRINT D$;"OPEN TCON"
30  PRINT D$;"WRITE TCON"
100  PRINT "0  D$ = CHR$(4) : PRINT D$;"Q$;"
     OPEN FILE";Q$; CHR$ (13)
110  PRINT "1  PRINT D$;"Q$;"WRITE FILE";Q$;
     ": LIST 3-"; CHR$ (13)
120  PRINT "2  PRINT D$;"Q$;"CLOSE FILE";Q$;
     ": END"; CHR$ (13)
130  PRINT "RUN"
140  PRINT "0"; CHR$ (13): PRINT "1"; CHR$ (
     13): PRINT "2"; CHR$ (13)
```

©

Apple Fast Sort

John Sarver

It can take a long time to put a list into alphabetical order. In a recent experiment, using a basic bubble sort routine, it took the author's Apple eight hours and 57 minutes to sort 1000 randomly created strings of random length between one and 20 characters. This subroutine puts both one- and two-dimensional Apple arrays in order at a tolerable speed: that same list of 1000 strings now takes one minute and 45 seconds.

Strings values, when assigned, are stored at the very top of Apple's free RAM, and as more strings are assigned, they are stored below the strings already in memory. A table, created when you use the DIM statement, keeps track of where each string is in RAM.

Some important information is stored at the beginning of this table. The first byte represents the first character in the variable name. The second byte represents the second character in the variable name plus $80 (adding $80 designates it as a string array rather than an integer or decimal point number array). The next pair of bytes gives the length of this pointer table.

The fifth byte is the number of dimensions that you have used with the DIM statement. If you used a two-dimensional array, the next two bytes tell how many variables are in the second part of the dimension (if three-dimensional, the next four bytes, and so on).

The final two bytes of information are the number of strings in the first dimension. The table begins here. Each variable is located by a three-byte pointer. The first byte is the length of the record, and the next two point to where the first character of the variable is stored. These pointers are always in order from the zero dimension to the nth dimension.

At the end of this grouping of pointers are the pointers for the first group of the second dimensioned part of the array. Following this is the second group of pointers for the second dimensioned part of the array, and so on. If you used a one-dimensional array, there is only one group of pointers.

As you can see, there is no need to sort the strings themselves. Just sort the pointers. Therefore, there is no time wasted in garbage collection and, in most cases, the length of the strings does not affect the time of execution.

Simple To Use

Using this sort is quite simple. Apple stores the last variable used in $81 and $82, so you may need to insert a statement in your BASIC program such as A$(0) = A$(0) (see line 90 of Program 2), or you may POKE these values in if you are putting this utility on another machine. The sort can be easily changed to use the zero dimension of an array if you wish. To do this, simply change the following lines in the BASIC loader (Program 1).

```
120  IF CK <  > 56854 THEN PRINT "CHECK DAT
     A STATEMENTS FOR ERROR": STOP
200  DATA  169,0,133,253,133,239,169,1
400  DATA  165,6,105,2,133,6,169,0
```

If you are using a two-dimensional array, you will need to store the records that are to be put in order by using the zero subscript of the second dimension (that is, A$(1,0), A$(2,0), etc.). The accompanying arrays (A$(1,1), A$(2,1), A$(1,2), A$(2,2), etc.) will be kept with their respective zero-subscripted record.

The sort will automatically ascertain if you are using a one- or two-dimensional array and will adjust itself accordingly. You may use any number of subscripts desired in one-dimensional arrays and in the first part of the two-dimensional arrays. But don't try to use anything larger than a two-dimensional array, or attempt to use more than 255 variables in the second part of your two-dimensional array. Some of the corresponding subarrays would not be properly aligned.

Program 1 loads the machine language sorting routine into RAM. You should save this on disk by typing:

BSAVE SORT, A$944A,L$1B6

Program 2 provides an example of the steps necessary to use the routine.

Program 1: ML Fast Sort Loader

```
100  REM  THIS PROGRAM INSTALLS BUT DOES
      NOT RUN THE ML FAST SORT
110  FOR I = 37962 TO 38399: READ A:CK =
      CK + A: POKE I,A: NEXT
120  IF CK <  > 56857 THEN  PRINT "CHECK
      DATA STATEMENTS FOR ERROR": STOP
130  TEXT : HOME : PRINT "TYPE 'BSAVE SORT,
      A $944A,L$1B6'"
140  PRINT "TO SAVE SORT ROUTINE ON DISK"
```

```
150  NEW
200  DATA   169,0,133,253,169,1,133,239
210  DATA   133,31,166,107,134,6,166,108
220  DATA   134,7,165,129,160,0,209,6
230  DATA   208,3,32,126,148,200,208,246
240  DATA   232,134,7,228,112,208,239,209
250  DATA   6,208,3,32,126,148,200,196
260  DATA   111,208,244,96,165,130,200,208
270  DATA   2,230,7,209,6,240,10,192
280  DATA   0,208,2,198,7,136,165,129
290  DATA   96,192,0,208,2,198,7,136
300  DATA   24,152,101,7,133,7,169,0
310  DATA   101,7,133,7,104,104,56,160
320  DATA   4,177,6,233,1,240,8,200
330  DATA   200,177,6,133,31,169,2,24
340  DATA   101,6,105,5,133,6,169,0
350  DATA   101,7,133,7,160,0,177,6
360  DATA   133,249,133,251,133,26,200,177
370  DATA   6,133,250,133,25,162,2,24
380  DATA   165,250,101,25,133,25,165,251
390  DATA   101,26,133,26,202,208,240,24
400  DATA   165,6,105,5,133,6,169,0
410  DATA   101,7,133,7,56,165,250,229
420  DATA   239,133,250,133,252,176,10,165
430  DATA   239,240,6,198,249,165,249,133
440  DATA   251,165,6,133,237,165,7,133
450  DATA   238,169,0,198,250,197,250,208
460  DATA   42,197,249,240,5,198,249,24
470  DATA   144,33,197,253,240,18,133,253
480  DATA   198,252,165,251,133,249,165,252
490  DATA   133,250,208,213,165,251,208,1
500  DATA   96,56,233,1,133,249,133,251
510  DATA   24,144,198,24,165,237,133,235
520  DATA   105,3,133,237,165,238,133,236
530  DATA   105,0,133,238,160,0,132,254
540  DATA   177,235,208,6,177,237,240,177
550  DATA   208,54,209,237,240,8,144,6
560  DATA   177,237,240,165,133,254,133,255
570  DATA   162,0,200,177,235,149,0,177
580  DATA   237,149,2,232,192,2,208,242
590  DATA   160,0,177,0,209,2,240,4
600  DATA   144,135,176,12,200,196,255,208
610  DATA   241,165,254,208,3,76,19,149
620  DATA   169,1,133,253,160,0,177,235
630  DATA   72,177,237,145,235,104,145,237
640  DATA   200,192,3,208,241,166,31,202
650  DATA   240,45,24,165,235,101,25,133
660  DATA   27,165,236,101,26,133,28,165
670  DATA   237,101,25,133,29,165,238,101
680  DATA   26,133,30,160,0,177,27,72
690  DATA   177,29,145,27,104,145,29,200
700  DATA   192,3,208,241,202,208,3,76
710  DATA   19,149,24,165,27,101,25,133
720  DATA   27,165,28,101,26,133,28,165
730  DATA   29,101,25,133,29,165,30,101
740  DATA   26,133,30,24,144,205,141,183
```

Program 2: Steps Necessary To Use Fast Sort

```
10  HIMEM: 37962
20 D$ =  CHR$ (4)
30  PRINT D$"BLOAD SORT"
40  INPUT "HOW MANY RECORDS";N
45  DIM A$(N)
50  FOR A = 1 TO N
60  PRINT "WHAT IS RECORD #"A;
70  INPUT " ";A$(A)
80  NEXT
90 A$(0) = A$(0)
100  CALL 37962
110  FOR A = 1 TO N
120  PRINT A$(A)
130  NEXT
140  END
```

©

VERSACALC
VERSACALC
VERSACALC
VERSACALC
VERSACALC
VERSACALC
VERSACALC
VERSACALC
VERSACALC
VERSACALC

TUTORIAL
UTILITIES
AND FILE MANAGER

Everything you always wanted to do with Visicalc,
(but thought you couldn't).

YES! We said "SORT VISICALC"!

NOW YOU CAN:

* SORT a Visicalc screen on any column, ascending or descending; all related formulas and labels are sorted too.
* put the entire disk CATALOG on the screen at once!
* easily do Year-To-Date accumulations!
* "pound" formulas to expose the full formulas in place on the screen![1]
* append two Visicalc files![1]
* print the contents of a /SS file!
* print the contents of a /PF file!
* AND our EASEL BINDER is so nice that you will put your other manual in it!

Apple II*	$100
PET & CBM*[1]	125
IBM PC[1]	150
Apple III[1]	150

*specify DOS

If you use Visicalc™ but you are bumping into its limitations, then you need Versacalc™! Versacalc runs within Visicalc but uses no extra memory; in fact, it effectively increases memory by letting you call in modules from disk as needed.

A Tutorial section makes clear such features as @LOOKUP, DIF, @NA, @ERROR, which are not well explained in the Visicalc manual.

A Utilities section makes it easy to create your own menu-driven modules which condense hundreds of commands into four keystrokes. You can build in sophisticated error checking (e.g. Is the input value between certain limits?). Now it is possible for people untrained in Visicalc to perform the weekly updating without constant instruction.

Anthro-Digital Software
P.O. Box 1385
Pittsfield, MA 01202
413-448-8278

Apple II is a trademark of Apple Computer, Inc.
Versacalc is a trademark of Versacalc Enterprises, Inc.
Visicalc is a trademark of Visicorp Inc.

The Apple Hi-Res Painter

James Totten

"Hi-Res Painter" is a graphics editor for use with a 32K Apple. With it you can: use any one of six colors (or combine colors with your "pen"); select from three different drawing pens; label pictures with upper- and lowercase lettering; color in squares, rectangles; and more.

When using the Apple's hi-res graphics, it seems that a lot of work can yield few results. This is true, of course, only if you are doing your graphics manually (HPLOT 0,0 TO 45,67 etc.). Since I use the graphics considerably (they are one reason I bought the computer), I didn't enjoy taking hours to draw a fairly impressive title page or chart or some other type of picture.

Menu Options

The "Hi-Res Painter" runs from four menus: Main Menu (1), Accessory Menu (2), Diskette Menu (3), and, most important of all, the Picture Menu (4). When you start, you are automatically placed at the first menu (Main). From here you can select to go to any of the other three menus presented by just pressing the first letter of its name. This letter is highlighted on the screen.

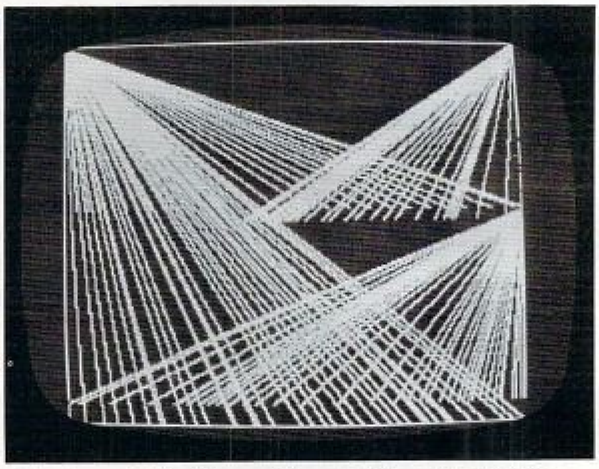
Pressing *A* will take you to the Accessory Menu (2). Here, you can choose from p)rint, f)ill, k)eyboard, and m)ain. The print option will work for those who own either a Trendcom or Silentype printer only. The *fill* option works for everyone. You select two points on the screen: the first is the upper left corner of the square you wish filled, and the other is the lower right corner. Presto! The keyboard option allows the user to change from paddle or joystick control of the pen to keyboard control of the pen. With the change, the I, J, K, M keys move the pen in the direction they are positioned. And, of course, the main option will take you to the main menu again.

The next menu in the list is the Diskette Menu, number three, and you can call that menu by pressing *D*. Here you can n)ame, d)elete, s)ave, l)oad, or r)ename any picture – s)ave will save the picture currently on the screen. Again, m)ain will return you to menu 1.

Finally, menu four is the Picture Menu, and to call it up press *P*. The available options here are: v)iew, l)abel, b)drop, c)olor, d)raw, e)rase, p)ens, and m)ain. The first option allows simply a total view (no text) of the graphics screen which you are working on. *Label* will do just that; you are asked for a date, name, or whatever to be typed in on the keyboard, and it is then transferred to a location of your choice onto the graphics screen.

The *b)drop* option stands for backdrop, and this will simply fill the screen (rather quickly) with a color of your choice. *Color* will allow you to choose a new color. Press the first letter of each as in the menu selections. *Draw* and *erase* are obvious in that they do exactly what they say. A note of warning though: if a picture is erased, it cannot be recalled unless it is on disk. The *pens* option is actually two in one. With it you can change the size of your pen (press 1, 2, or 3 and watch the screen), and turn it on or off. And again, main returns you to menu one. You can draw using paddles or a joystick, or you can switch the controls to use the keyboard.

To produce very good-looking designs, try some experiments. Fantastic pictures (such as stars on a moonlit night) can easily be created by just moving the pen in various sizes and colors.


A design created with a paddle controller using "Hi-Res Painter."

Program 1: Hi-Res Painter

```
20  LOMEM: 24576: ONERR  GOTO 1045
21  DIM PX(2),PY(2),C$(6),P$(1)
25  FOR L = 1 TO 4:MX(L) = 0:MY(L) = 0: NEXT
     L:D$ =  CHR$ (4):C = 3:P = 0:BC = 0
30 KI =  - 16384:RK =  - 16368:B0 =  - 16287
     :B1 =  - 16286:TG =  - 16301:FG =  - 16
     302
35 P$(0) = "OFF":P$(1) = "ON":C$(1) = "GREEN
     ":C$(2) = "PINK":C$(3) = "WHITE"
40 C$(4) = "BLACK":C$(5) = "ORANGE":C$(6) =
     "LT.BLUE":I = 1:P$ = "NOT NAMED"
41  IF  PEEK (233) <  > 64 THEN  PRINT D$"BL
     OAD CHARACTERS/SH2": POKE 232,0: POKE 2
     33,64
42  SCALE= 1: ROT= 0:X = 139:Y = 80
43  TEXT : HOME : NORMAL : VTAB 10: PRINT
     TAB(11)"THE HI-RES PAINTER": PRINT  TAB
     ( 7)"-=(                   )=-": PRINT
     TAB(11)"BY JAMES R. TOTTEN"
44  POKE RK,0: VTAB 24: PRINT "<< TO BEGIN P
```

```
     USH ANY KEY EXCEPT RESET >>"
45   IF  PEEK (KI) < 128 THEN 45
46   POKE RK,0
50   HGR : HCOLOR= C: POKE TG,0: POKE 34,20:
     HOME
55   PRINT "PAINTER MENU NUMBER 1 (MAIN)":
     PRINT
60   PRINT "A)CCESSORY   D)ISKETTE   P)ICTURE
       >";: GET K$
65   IF K$ =  CHR$ (27) THEN  POKE RK,0: POKE
      34,0: TEXT : HOME : END
70   IF K$ = "P" THEN 100
75   IF K$ = "A" THEN 450
80   IF K$ = "D" THEN 300
85   POKE RK,0: HOME : GOTO 55
100  POKE RK,0: HOME
105  PRINT "PAINTER MENU NUMBER 4 (PICTURE)"
     : PRINT
110  PRINT "V)IEW  L)ABEL  B)DROP  C)OLOR
      D)RAW  E)RASE  P)ENS   M)AIN  >";: GET K$
115  IF K$ = "M" THEN 85
120  IF K$ =  CHR$ (27) THEN  POKE RK,0: POKE
      34,0: TEXT : HOME : END
125  IF K$ = "E" THEN  HGR :BC = 0: GOTO 100
130  IF K$ = "V" THEN 145
132  IF K$ = "C" THEN 150
134  IF K$ = "B" THEN 240
136  IF K$ = "D" THEN 185
138  IF K$ = "P" THEN 164
140  IF K$ = "L" THEN 218
142  POKE RK,0: HOME : GOTO 105
145  POKE FG,0
146  IF  PEEK (KI) > 127 THEN  POKE TG,0:
      GOTO 100
147  GOTO 146
150  POKE RK,0: HOME : PRINT "CURRENT COLOR:
      ";: INVERSE : PRINT C$(C): NORMAL :
      PRINT
152  PRINT "G)REEN    O)RANGE    W)HITE
      B)LACK    L)T.BLUE  P)INK      >"; : GET K$
154  IF K$ = "G" THEN C = 1: GOTO 100
155  IF K$ = "P" THEN C = 2: GOTO 100
156  IF K$ = "W" THEN C = 3: GOTO 100
158  IF K$ = "B" THEN C = 4: GOTO 100
159  IF K$ = "O" THEN C = 5: GOTO 100
160  IF K$ = "L" THEN C = 6: GOTO 100
162  GOTO 150
164 XC =  INT ( PDL (0)):YC =  INT ( PDL (1) )
165  POKE RK,0: HOME : PRINT "PEN OPERATIONS
     ": PRINT
166  PRINT "S)ET CURSOR SIZE     T)URN ON/OFF
      >";: GET K$
167  IF K$ = "S" THEN 172
168  IF K$ <  > "T" THEN 165
169 P = P + 1: IF P > 1 THEN P = 0
170  HOME : PRINT : PRINT "PEN IS NOW "P$(P)
     : FOR L = 1 TO 300: NEXT L
171  GOTO 100
172  POKE RK,0: HOME : PRINT "TYPE A NUMBER
     FROM 1 TO 3 FOR CURSOR     SIZE (1=SMALL
     EST). CURSOR IS SHOWN ON    SCREEN. WHEN
      DONE, PUSH RETURN.  >";: GET K$
174  IF K$ =  CHR$ (13) THEN 100
176  IF K$ = "1" THEN CS = 0
177  IF K$ = "2" THEN CS = 4
178  IF K$ = "3" THEN CS = 8
179  HCOLOR= BC: FOR L = XC - 1 TO XC + 8:
     HPLOT L,YC - 1 TO L,YC + 8: NEXT L:
     HCOLOR= C
180  FOR L = XC TO XC + CS: HPLOT L,YC TO L,
     YC + CS: NEXT L
182  GOTO 172
185  IF K THEN 1010
186  POKE RK,0: HOME : PRINT : PRINT "TO BEG
     IN OR STOP DRAWING PUSH ANY KEY ";: GET
     K$
187  POKE FG,0: POKE RK,0
190  IF CS = 0 THEN LL = 1:RL = 279:TL = 0:B
     L = 191
191  IF CS = 4 THEN LL = 1:RL = 274:TL = 0:B
     L = 186
192  IF CS = 8 THEN LL = 1:RL = 270:TL = 0:B
     L = 182
194  HCOLOR= C
196 X =  INT ( PDL (0)):Y =  INT ( PDL (1))
198  IF X < LL THEN X = LL
200  IF X > RL THEN X = RL
202  IF Y > BL THEN Y = BL
204  FOR L = X TO X + CS: HPLOT L,Y TO L,Y +
     CS: NEXT L
205  IF  PEEK (KI) > 127 THEN  POKE TG,0: GOTO
     100
206  IF P THEN 210
208  HCOLOR= BC: FOR L = X TO X + CS: HPLOT
     L,Y TO L,Y + CS: NEXT L: HCOLOR= C
209  IF  PEEK (KI) > 127 THEN  POKE TG,0: GOTO
     100
210  IF CS = 0 THEN  IF  PEEK (B1) > 127 THEN
      CALL  - 198:XO = X:YO = Y
212  IF CS = 0 THEN  IF  PEEK (BO) > 127 THEN
      HPLOT X,Y TO XO,YO
215  GOTO 196
218  POKE RK,0: HOME : PRINT : INPUT "ENTER
     LABEL >";L$
219  IF L$ = "" THEN 218
220  HOME : PRINT : PRINT "DO YOU WANT IT ON
      TOP OR BOTTOM (T/B)? ";: GET K$
222  IF K$ = "B" THEN Y = 180: GOTO 226
224  IF K$ = "T" THEN Y = 6: GOTO 226
225  GOTO 220
226 L =  LEN (L$): IF L > 26 THEN 218
228 X = 137 -  INT ((L / 2) * 8)
230  FOR P = 1 TO L: IF  ASC ( MID$ (L$,P,1)
     ) < 62 THEN K =  ASC ( MID$ (L$,P,1)) -
     31: GOTO 232
231 K =  ASC ( MID$ (L$,P,1)) - 3
232  HCOLOR= 0: FOR L = X - 2 TO X + 7: HPLOT
     L,Y - 1 TO L,Y + 8: NEXT L: HCOLOR= 3
233  DRAW K AT X,Y:X = X + 8: NEXT P
234  HCOLOR= C: GOTO 100
240  POKE RK,0: HOME : PRINT "COLORS FOR BAC
     KDROP...": PRINT : PRINT "G)REEN  B)LUE
      P)INK  W)HITE  O)RANGE": PRINT ">";:
      GET K$
242  IF K$ = "G" THEN  HCOLOR= 1:BC = 1: GOTO
     248
243  IF K$ = "B" THEN  HCOLOR= 6:BC = 6: GOTO
     248
244  IF K$ = "P" THEN  HCOLOR= 2:BC = 2: GOTO
     248
245  IF K$ = "W" THEN  HCOLOR= 3:BC = 3: GOTO
     248
246  IF K$ = "O" THEN  HCOLOR= 5:BC = 5: GOTO
     248
247  GOTO 240
248  HPLOT 0,0: CALL 62454
250 BD = 1: GOTO 100
300  POKE RK,0: HOME
302  PRINT "PAINTER MENU NUMBER 3 (DISKETTE)
     ": PRINT
304  PRINT "N)AME  D)ELETE  S)AVE
      L)OAD  R)ENAME  M)AIN   >";: GET K$
306  IF K$ = "M" THEN 85
308  IF K$ =  CHR$ (27) THEN  POKE RK,0: POKE
     34,0: TEXT : HOME : END
310  IF K$ = "N" THEN 320
311  IF K$ = "S" THEN 335
312  IF K$ = "L" THEN 355
313  IF K$ = "R" THEN 385
314  IF K$ = "D" THEN 370
315  GOTO 300
320  POKE RK,0: HOME : PRINT "USE NO COMMAS
     OR COLONS IN NAME.": PRINT : INPUT "> "
     ;P$
```

```
325  IF P$ = "" THEN 320
330  HOME : PRINT "NAME: "P$: NORMAL
332  PRINT : PRINT "IS THIS CORRECT? ";: GET
     K$: IF K$ = "N" THEN 320
333  IF K$ = "Y" THEN 300
334  POKE RK,0: GOTO 330
335  IF P$ = "NOT NAMED" THEN  HOME : CALL  -
     198: POKE RK,0: PRINT : PRINT "PICTURE
     HAS NOT BEEN NAMED": FOR L = 1 TO 550:
     NEXT L: GOTO 300
340  POKE RK,0: HOME : PRINT "PICTURE NAME:
     "P$: PRINT
345  PRINT "SAVE WITH THIS NAME? ";: GET K$:
      PRINT K$: IF K$ = "Y" THEN 350
346  IF K$ = "N" THEN 300
347  GOTO 340
350  PRINT D$"BSAVE "P$",A$2000,L$1FFF": GOTO
     300
355  POKE RK,0: HOME : PRINT : INPUT "NAME?
     ";P$
356  IF P$ = "" THEN 355
358  HOME : PRINT "PICTURE NAME: "P$: PRINT
360  PRINT "IS THIS NAME CORRECT? ";: GET K$
     : PRINT K$
362  IF K$ = "N" THEN 300
363  IF K$ = "Y" THEN 365
364  GOTO 358
365  PRINT D$"BLOAD "P$
366  GOTO 300
370  POKE RK,0: HOME : PRINT : INPUT "NAME?
     ";P$
371  IF P$ = "" THEN 370
372  HOME : PRINT "PICTURE NAME: "P$: PRINT
375  PRINT "DELETE THIS PICTURE? ";: GET K$:
      PRINT K$
376  IF K$ = "Y" THEN 380
377  IF K$ = "N" THEN 300
378  GOTO 372
380  PRINT D$"DELETE "P$: GOTO 300
385  POKE RK,0: HOME : PRINT "USE NO COMMAS
     OR COLONS IN NEW NAME": PRINT
388  INPUT "CURRENT NAME? ";P1$: IF P1$ = ""
      THEN 385
390  INPUT "NEW NAME? ";P2$: IF P2$ = "" THEN
     385
393  HOME : PRINT "OLD NAME: "P1$: PRINT "NE
     W NAME: "P2$: PRINT
395  PRINT "ARE THESE BOTH CORRECT? ";: GET
     K$: PRINT K$: IF K$ = "N" THEN 385
396  IF K$ = "Y" THEN 400
398  GOTO 393
400  PRINT D$"RENAME "P1$","P2$: GOTO 300
450  POKE RK,0: HOME
452  PRINT "PAINTER MENU NUMBER 2 (ACCESSORY
     )": PRINT
454  PRINT "P)RINT  F)ILL  K)EYBOARD  M)AIN
        >";: GET K$
456  IF K$ = "M" THEN  POKE RK,0: HOME : GOTO
     55
458  IF K$ =  CHR$ (27) THEN  TEXT : POKE RK
     ,0: HOME : END
459  IF K$ = "P" THEN 475
460  IF K$ = "F" THEN 500
461  IF K$ = "K" THEN 465
462  GOTO 450
465  POKE RK,0: HOME : IF K THEN K = 0: GOTO
     468
466  IF  NOT K THEN K = 1
468  IF K = 0 THEN  PRINT : PRINT "KEYBOARD
     IS OFF"
469  IF K = 1 THEN  PRINT : PRINT "KEYBOARD
     IS ON"
470  FOR L = 1 TO 300: NEXT L: GOTO 450
475  POKE RK,0: HOME : PRINT "PICTURE PRINTI
     NG OPTIONS -": PRINT
476  PRINT "I)NVERSED  N)ORMAL
        R)OTATED   C)ONTINUE    >";: GET K$
478  IF K$ = "N" THEN ST = 0: GOTO 475
480  IF K$ = "I" THEN ST = 1: GOTO 475
482  IF K$ = "R" THEN RR = 1: GOTO 475
484  IF K$ = "C" THEN 488
486  GOTO 475
488  POKE RK,0: HOME : PRINT : PRINT "TURN P
     RINTER ON AND PRESS ANY KEY ";: GET K$
490  IF RR AND ST THEN  POKE 1145,88: CALL  -
     16038: GOTO 450
492  IF RR THEN  POKE 1145,120: CALL  - 1603
     8: GOTO 450
494  IF ST THEN  POKE 1400,0: CALL  - 16036:
      GOTO 450
496  CALL  - 16044: GOTO 450
500  POKE RK,0: HOME : INPUT "UPPER LEFT POI
     NT (X,Y)  >";UX$,UY$: IF UX$ = "" OR UY
     $ = "" THEN 500
505  IF ( VAL (UX$) < 0) OR ( VAL (UX$) > 27
     9) THEN 500
506  IF ( VAL (LY$) < 0) OR ( VAL (LY$) > 19
     1) THEN  VTAB  PEEK (37): GOTO 507
507  INPUT "LOWER RIGHT POINT (X,Y)  >";LX$,
     LY$: IF LX$ = "" OR LY$ = "" THEN  VTAB
      PEEK (37): GOTO 507
508  IF ( VAL (LX$) < 0) OR ( VAL (LX$) > 27
     9) THEN  VTAB  PEEK (37): GOTO 507
510  HOME : PRINT : PRINT "PRESS A KEY TO BE
     GIN FILL ";: GET K$: PRINT K$
511  HCOLOR= C
515  FOR L =  VAL (UX$) TO  VAL (LX$): HPLOT
     L, VAL (UY$) TO L, VAL (LY$): NEXT L
520  GOTO 450
1010  POKE RK,0: HOME : PRINT : PRINT "TO BE
     GIN OR STOP DRAWING PUSH RETURN ";: GET
     K$
1012  POKE FG,0: POKE RK,0
1015  IF CS = 0 THEN LL = 1:RL = 279:TL = 0:
     BL = 191
1016  IF CS = 4 THEN LL = 1:RL = 274:TL = 0:
     BL = 186
1017  IF CS = 8 THEN LL = 1:RL = 270:TL = 0:
     BL = 182
1018  HCOLOR= C
1019  FOR L = X TO X + CS: HPLOT L,Y TO L,Y +
     CS: NEXT L
1020  IF  NOT P THEN  HCOLOR= BC: FOR L = X TO
     X + CS: HPLOT L,Y TO L,Y + CS: NEXT L:
     HCOLOR= C
1021  IF  PEEK (KI) < 128 THEN 1019
1023 L =  PEEK (KI)
1024  IF L = 201 THEN Y = Y - 1: GOTO 1036
1025  IF L = 205 THEN Y = Y + 1: GOTO 1036
1026  IF L = 202 THEN X = X - 1: GOTO 1036
1027  IF L = 203 THEN X = X + 1: GOTO 1036
1028  IF L = 213 THEN X = X - 1:Y = Y - 1:
     GOTO 1036
1029  IF L = 206 THEN X = X - 1:Y = Y + 1:
     GOTO 1036
1030  IF L = 207 THEN X = X + 1:Y = Y - 1:
     GOTO 1036
1031  IF L = 172 THEN X = X + 1:Y = Y + 1:
     GOTO 1036
1032  IF (CS = 0) AND (L = 211) THEN XO = X:
     YO = Y: CALL  - 198: GOTO 1036
1033  IF (CS = 0) AND (L = 196) THEN  HPLOT
     X,Y TO XO,YO: GOTO 1036
1034  IF L = 141 THEN  POKE TG,0: GOTO 100
1035  POKE RK,0: GOTO 1021
1036  IF X < LL THEN X = LL
1037  IF X > RL THEN X = RL
1038  IF Y > BL THEN Y = BL
1039  IF Y < TL THEN Y = TL
1040  POKE RK,0: GOTO 1019
1045  HOME : PRINT : PRINT "DISK ERROR CODE
     " PEEK (222): PRINT "CHECK SYNTAX AND T
     RY AGAIN >";: GET K$
1050  POKE RK,0: HOME : GOTO 55
```

Program 2: Shape Table For Picture Labels

```
4000- 58 00 B2 00 C5 00 D8 00
4008- EC 00 02 01 15 01 29 01
4010- 3C 01 4F 01 62 01 75 01
4018- 8A 01 9D 01 B0 01 C3 01
4020- D6 01 E9 01 FE 01 12 02
4028- 26 02 3B 02 50 02 65 02
4030- 79 02 8D 02 A2 02 B6 02
4038- C9 02 DD 02 F1 02 06 03
4040- 19 03 2C 03 41 03 55 03
4048- 69 03 7D 03 91 03 A5 03
4050- B8 03 CC 03 DF 03 F2 03
4058- 06 04 19 04 2C 04 40 04
4060- 54 04 68 04 7C 04 8F 04
4068- A3 04 B6 04 C9 04 DD 04
4070- F1 04 05 05 1A 05 2E 05
4078- 41 05 54 05 67 05 7C 05
4080- 90 05 A3 05 B7 05 CC 05
4088- E0 05 F4 05 08 06 1C 06
4090- 30 06 43 06 57 06 6B 06
4098- 7F 06 94 06 A8 06 BC 06
40A0- D0 06 E4 06 F8 06 0D 07
40A8- 21 07 36 07 4B 07 5F 07
40B0- 74 07 49 09 1A 1B 1B 4A
40B8- 49 1A 1B 1B 4A 49 1A 1B
40C0- 1B 4A 49 02 00 09 4D 1A
40C8- 1B 1F 4A 4D 1A 1B 1F 4A
40D0- 4D 1A 1B 1B 4A 4D 02 00
40D8- 69 0D 1A 3B 3B 0A 0D 0D
40E0- 1A 1B 1B 4A 49 1A 1B 1B
40E8- 4A 49 02 00 69 0D 1A 3B
40F0- 3B 2A 2D 2D 1A 3B 3B 2A
40F8- 2D 2D 1A 3B 3B 0A 0D 0D
4100- 02 00 09 4D 1A 3F 3F 6A
4108- 4D 1A 3B 3F 4A 0D 15 1B
4110- 3F 77 69 11 00 6D 09 1A
4118- 1F 3B 4E 69 1A 1B 1F 0A
4120- 4D 11 3B 1F 73 09 2D 02
4128- 00 69 09 1A 1B 1F 6E 4D
4130- 1A 1B 3B 6A 0D 15 1B 1F
4138- 73 6D 15 00 49 0D 1A 1B
4140- 1F 0A 4D 11 1B 1B 53 49
4148- 11 1B 1B 53 49 11 00 09
4150- 4D 1A 1B 3B 6A 49 1A 1B
4158- 1B 6E 49 1A 1B 3B 4A 4D
4160- 02 00 09 4D 1A 3B 1B 4A
4168- 09 15 3B 1B 53 49 15 1B
4170- 1F 53 69 11 00 09 4D 1A
4178- 1F 1F 0E 2D 0D 1A 3F 3F
4180- 0E 2D 0D 1A 1F 1F 4E 4D
4188- 02 00 49 09 1A 1B 1F 4A
4190- 4D 1A 3F 3F 4E 4D 1A 1B
4198- 1F 4A 49 02 00 49 09 1A
41A0- 1B 1B 4A 49 1A 1B 1B 4A
41A8- 6D 1A 3B 1F 0A 6D 11 00
41B0- 49 09 1A 1B 1B 4A 49 1A
41B8- 3F 3F 4E 49 1A 1B 1B 4A
41C0- 49 02 00 49 09 1A 1B 1B
41C8- 4A 49 1A 1B 1B 4A 49 1A
41D0- 1B 3F 0A 6D 11 00 49 09
41D8- 1A 1F 1B 4A 69 1A 1B 1F
41E0- 0A 4D 11 1B 1B 73 49 11
41E8- 00 29 6D 1A 1F 1B 6E 29
41F0- 15 3B 3B 33 6D 29 1A 1F
41F8- 1B 0E 2D 0D 02 00 09 4D
4200- 1A 1B 3F 4A 4D 1A 1B 1F
4208- 4A 4D 1A 1B 1F 0A 2D 0D
4210- 02 00 29 6D 1A 1F 1B 4E
4218- 09 15 1B 3F 53 4D 11 1B
4220- 1B 33 2D 2D 15 00 2D 2D
4228- 15 3B 1B 53 09 0D 1A 3B
4230- 1F 4A 09 15 3B 1B 73 2D
4238- 0D 02 00 49 0D 1A 3B 1F
4240- 0A 0D 0D 1A 3B 1B 2E 2D
4248- 2D 1A 3B 1B 4A 69 02 00
4250- 2D 2D 15 1B 1B 33 2D 6D
4258- 1A 1F 1B 4A 09 15 3B 1B
4260- 73 2D 0D 02 00 29 6D 1A
4268- 1F 1B 6E 49 1A 3B 3F 6E
4270- 09 15 3B 1B 73 2D 0D 02
4278- 00 2D 2D 15 3B 1B 53 09
```

```
4280- 0D 1A 1B 1F 0A 4D 11 1B
4288- 1B 57 4D 11 00 29 6D 1A
4290- 1F 1B 6E 09 15 1B 3F 17
4298- 4D 29 1A 1F 1B 0E 2D 0D
42A0- 02 00 29 6D 1A 1F 1B 6E
42A8- 09 15 3B 3F 57 49 15 3B
42B0- 1B 73 2D 0D 02 00 49 09
42B8- 1A 1B 3F 0A 6D 11 1B 1B
42C0- 53 6D 11 1B 3B 57 49 11
42C8- 00 49 09 1A 1B 3F 0A 6D
42D0- 11 1B 1B 53 6D 11 1B 3B
42D8- 17 6D 09 02 00 49 2D 1A
42E0- 3B 1F 0A 6D 11 1B 1B 77
42E8- 6D 11 1B 3F 53 09 2D 02
42F0- 00 49 09 1A 1B 1B 0A 2D
42F8- 0D 1A 1B 1B 0A 2D 0D 1A
4300- 1B 1B 4A 49 02 00 6D 09
4308- 1A 1B 3F 4A 6D 1A 3F 1B
4310- 4A 6D 1A 1B 3F 2A 4D 11
4318- 00 29 6D 1A 1F 1B 4E 09
4320- 15 1B 3F 53 4D 11 1B 1B
4328- 53 4D 11 00 29 6D 1A 1F
4330- 1B 6E 0D 15 3B 3F 33 0D
4338- 0D 15 1B 1B 73 2D 2D 02
4340- 00 49 09 1A 3B 3F 4A 09
4348- 15 3B 3F 17 4D 29 1A 3F
4350- 3F 4A 49 02 00 4D 09 1A
4358- 3B 1F 2E 4D 15 3B 1B 33
4360- 6D 29 1A 3B 1F 4E 49 02
4368- 00 49 09 1A 3B 3F 6A 09
4370- 15 1B 1B 33 4D 29 1A 3B
4378- 3F 4A 49 02 00 49 29 1A
4380- 1F 3F 6A 29 15 3B 1B 33
4388- 4D 2D 1A 1F 3F 4A 49 02
4390- 00 49 09 1A 3B 3F 6A 09
4398- 15 3B 3F 37 4D 09 1A 3B
43A0- 3F 4A 49 02 00 09 6D 1A
43A8- 1F 3B 0A 4D 11 1B 3B 77
43B0- 4D 11 1B 1B 57 49 11 00
43B8- 49 09 1A 1F 3F 6A 29 15
43C0- 3B 1F 73 6D 15 3B 1B 53
43C8- 2D 0D 02 00 4D 09 1A 1B
43D0- 1B 6E 6D 1A 1F 3B 6E 09
43D8- 15 3B 1B 73 49 11 00 09
43E0- 4D 1A 1B 1B 0A 6D 11 1B
43E8- 3B 53 69 11 1B 3F 57 49
43F0- 11 00 49 29 1A 1B 1B 4A
43F8- 29 15 3B 1B 53 49 15 3B
4400- 1B 73 2D 0D 02 00 4D 09
4408- 1A 3B 1B 6E 4D 1A 1B 3B
4410- 6E 4D 1A 3B 1B 4E 49 02
4418- 00 29 4D 1A 1B 1F 4A 4D
4420- 1A 1B 1F 4A 4D 1A 3B 3F
4428- 4A 49 02 00 49 09 1A 3B
4430- 3B 6A 0D 15 3B 3B 33 0D
4438- 0D 15 3B 3B 73 49 11 00
4440- 49 09 1A 3B 1F 2E 4D 15
4448- 3B 1B 33 4D 29 1A 1F 1B
4450- 4E 49 02 00 49 09 1A 3B
4458- 3F 6A 09 15 3B 1B 33 4D
4460- 29 1A 3B 3F 4A 49 02 00
4468- 49 09 1A 3B 1F 2E 4D 15
4470- 3B 1B 37 0D 6D 1A 1B 1B
4478- 6E 49 02 00 49 09 1A 1F
4480- 3F 6A 29 15 3B 1F 73 6D
4488- 15 3B 1B 53 49 15 00 49
4490- 09 1A 3B 1F 2E 4D 15 1B
4498- 1B 33 4D 09 1A 1B 1B 4E
44A0- 49 02 00 49 09 1A 3F 3F
44A8- 6A 49 1A 3B 3F 4A 09 15
44B0- 1B 3F 77 49 11 00 69 09
44B8- 1A 1B 3F 0E 4D 11 1B 1B
44C0- 57 4D 15 1B 3F 53 49 11
44C8- 00 49 09 1A 1F 1B 6E 09
44D0- 15 3B 1B 33 4D 2D 1A 1F
44D8- 3F 4A 49 02 00 49 09 1A
44E0- 1F 1B 6E 09 15 1B 1F 57
44E8- 0D 0D 1A 1B 1F 4A 49 02
44F0- 00 49 09 1A 1F 1F 6E 0D
44F8- 15 3B 3B 33 0D 0D 15 1B
```

```
4500- 1F 57 49 11 00 49 09 1A
4508- 1F 1B 0E 0D 0D 1A 1B 1F
4510- 0A 0D 0D 1A 1F 1B 4E 49
4518- 02 00 49 09 1A 1F 1B 6E
4520- 09 15 3B 1F 73 6D 15 3B
4528- 1B 53 2D 0D 02 00 49 09
4530- 1A 3F 3F 4E 69 1A 1B 1F
4538- 0A 4D 11 3B 3F 77 49 11
4540- 00 29 4D 1A 3B 1B 4A 69
4548- 1A 1F 1B 4A 69 1A 3B 1B
4550- 0A 6D 11 00 09 4D 1A 3B
4558  3D 6A 07 15 1D 1D 53 49
4560- 11 1B 1B 53 49 11 00 09
4568- 4D 1A 3B 3B 6A 09 15 3B
4570- 1B 33 2D 2D 15 3B 1B 33
4578- 4D 29 02 00 2D 6D 1A 1F
4580- 3B 0A 4D 15 1B 3F 57 4D
4588- 15 3B 1B 17 2D 6D 02 00
4590- 09 6D 1A 1F 3B 6A 49 1A
4598- 1B 1B 6E 49 1A 1F 3B 4A
45A0- 6D 02 00 2D 6D 1A 1F 3B
45A8- 0A 4D 15 3B 1B 57 4D 15
45B0- 3B 1B 17 2D 6D 02 00 2D
45B8- 2D 15 3B 1B 33 4D 09 1A
45C0- 1B 3F 6E 49 1A 1F 1B 2E
45C8- 2D 2D 02 00 2D 2D 15 3B
45D0- 1B 33 4D 09 1A 1B 3F 6E
45D8- 49 1A 1B 1B 6E 49 02 00
45E0- 29 6D 1A 1F 1B 6E 49 1A
45E8- 3F 1F 6E 09 15 3B 1B 73
45F0- 2D 0D 02 00 4D 29 1A 1F
45F8- 1B 6E 09 15 3B 3F 37 4D
4600- 29 1A 1F 1B 6E 09 15 00
4608- 29 6D 1A 1B 1F 4A 4D 1A
4610- 1B 1F 4A 4D 1A 1B 1F 0A
4618- 2D 0D 02 00 09 2D 15 1B
4620- 1F 53 09 0D 1A 3B 1B 4A
4628- 69 1A 3B 1B 0E 6D 11 00
4630- 4D 29 1A 3B 1B 6E 4D 1A
4638- 1B 3B 6E 4D 1A 3B 1B 6E
4640- 09 15 00 6D 09 1A 1B 3B
4648- 0A 4D 11 1B 1B 57 4D 11
4650- 3B 1B 17 2D 2D 15 00 4D
4658- 29 1A 3F 3B 6E 0D 15 3B
4660- 1B 33 4D 29 1A 1F 1B 6E
4668- 09 15 00 4D 29 1A 1F 1B
4670- 2E 4D 15 3B 3B 33 4D 2D
4678- 1A 1F 1B 6E 09 15 00 29
4680- 6D 1A 1F 1B 6E 09 15 3B
4688- 1B 33 4D 29 1A 1F 1B 0E
4690- 2D 0D 02 00 2D 6D 1A 1F
4698- 3B 0A 4D 15 1B 3F 57 4D
46A0- 11 1B 1B 17 6D 09 02 00
46A8- 29 6D 1A 1F 1B 6E 09 15
46B0- 3B 1B 33 0D 0D 15 1B 1F
46B8- 73 6D 15 00 2D 6D 1A 1F
46C0- 3B 0A 4D 15 1B 3F 57 0D
46C8- 0D 1A 1F 3B 2A 4D 15 00
46D0- 29 6D 1A 1F 1B 6E 49 1A
46D8- 3B 3F 4A 09 15 3B 1B 73
46E0- 2D 0D 02 00 2D 2D 15 3B
46E8- 3B 73 69 11 1B 3B 53 69
46F0- 11 1B 3B 53 2D 0D 02 00
46F8- 4D 29 1A 1F 1B 6E 09 15
4700- 3B 1B 33 4D 29 1A 1F 1B
4708- 0E 2D 0D 02 00 4D 29 1A
4710- 1F 1B 6E 09 15 3B 1B 33
4718- 4D 29 1A 3B 3B 4A 4D 02
4720- 00 4D 29 1A 1F 1B 6E 09
4728- 15 3B 1B 33 0D 0D 15 3B
4730- 1F 37 4D 29 02 00 4D 29
4738- 1A 1F 1B 0E 0D 0D 1A 1B
4740- 1F 0A 0D 0D 1A 1F 1B 6E
4748- 09 15 00 4D 29 1A 1F 1B
4750- 0E 0D 0D 1A 1B 1F 4A 4D
4758- 1A 1B 1F 4A 4D 02 00 2D
4760- 2D 15 3B 1B 73 09 0D 1A
4768- 1B 1F 0A 4D 11 3B 1B 33
4770- 2D 2D 15 00 2D 2D 15 3B
4778- 3F 37 2D 2D 15 3B 3F 37
4780- 2D 2D 15 3B 3F 37 2D 2D
4788- 15 00 00
```

The Printographer Graphics Printer Package For The Apple

Richard Cornelius

The Printographer, by Stephen Billard, is a utility program which processes high-resolution images on the Apple II computer and sends them to a printer. You can perform cropping operations on the high-resolution screen and determine the format in which the image will appear on the printer. The package comes with one disk including instructions on how to make backup copies and a 27-page manual.

Operates With Any Printer

The first question that many people will ask about *The Printographer* is, "Will it work with my printer?" The answer is yes, indeed, if your printer has any graphics capability. A powerful feature of the program is the ease with which it can be configured to operate with just about any combination of printer and interface.

If you have one of more than a dozen common printers, you do not need to know any technical details of its operation. From a menu, you can identify your printer and, if applicable, the particular interface card that you have. This menu automatically appears the first time you boot the disk. The printer specifications that you select are then saved to the disk so that on subsequent runs you are moved directly into the main program without having to identify your particular printer again.

The manual explains how to rerun the printer-selection program should you wish to run *The Printographer* with a different printer. If your printer is not one of those on the menu, then you explain how your printer processes information, but the program on the disk still does most of the work.

The Printographer performs its various cropping operations on the high-resolution screen quickly and smoothly. Pictures can be cropped from the top, bottom, or either side, or in a diamond or oval shape within the boundaries you specify. Starting over again is accomplished by a single keystroke, and the mechanics of operating the program are easy to understand.

However, I did encounter a problem. The cropping instructions are given on the text page and include, logically enough, the use of the question mark to return the instructions to the screen. The first time through the instructions, I paid little attention to them except to remember the use of the question mark.

Unfortunately, once I was on the high-resolution page for cropping, the question mark generated only a beep from the computer; I saw no instructions. Eventually I found that a CTRL-C would send the program to a point from which I could return to the instructions, but the first time through I had to reboot the disk just to see the instructions again.

Easy To Use

Most of the program is very easy to use. The general format employs the ESC key to move a highlighting identifier through the menu and the RETURN key to actually select the item that is highlighted. I was impressed with how easy it was to select a high-resolution picture from among a mixture of Applesoft, text, and binary files on one of my own disks. *The Printographer* gave me a menu of only the high-resolution images on my disk. It even ignored other binary files that were not high-resolution pictures.

The printing routines seem to work exactly as specified. Pictures can be printed in normal or inverse mode, vertically or horizontally on the page, magnified up to nine times, and tabbed over on the page. Routines for doing this printing from your own programs are available (not copy-protected) on the disk with instructions in the documentation on how to use them. Images can also be saved on a disk in one of three forms: a regular binary file, a compressed version that saves space, or a printer image. If you have the right printer setup, this last form allows printer spooling so that the computer is not tied up while the graphics are being printed.

The documentation is clear and complete. It is not packaged in a fancy (and expensive) padded binder, but it contains all of the information that I would want to know about the software. The primary part of the documentation is written so that no technical knowledge of the Apple or printers is required. The appendices, however, contain technical details such as writing your own printer driver and memory management so you can use some of the *Printographer* routines within your own programs.

The backup procedure seems to work well. Parts of the disk are copy-protected. The disk, however, comes with its own copying program which, according to the documentation, will make a total of three backup copies of the disk. This copy program uses a single drive, supposedly for assuring maximum reliability during the copying process. The copy program works essentially like *COPYA* on the Apple System

Master Disk.

All in all, *The Printographer* is a useful utility which is relatively easy to use. It comes with complete documentation. The program is not without faults, but technical support is easy to obtain. The price seems in line with the capabilities of the program. Its strongest feature is the manner in which it can easily be configured to work with whichever graphics printer you might happen to have.

The Printographer
Southwestern Data Systems
10761-E Woodside Avenue
Santee, CA 92071
$49.95

Marathon For Atari

Mike Kinnamon

More Than A Math Drill

Marathon departs somewhat from typical math drill programs. It makes use of the Atari's graphics and sound capabilities. The object is to advance your marathon runner from the starting line at the left of the screen to the finish line at the right. This is accomplished by giving the correct answer to the math problem presented before your opponent does or before the timer runs out.

When the game is loaded from disk or cassette, an option allows one or two players. You may then select which mathematical operation the game will focus on – addition, subtraction, multiplication, or division. You may opt for a mixture of these operations by selecting the general category instead.

Four Levels Of Difficulty

There are four levels of difficulty: walker, jogger, sprinter, and olympian. Each successively higher level decreases the amount of time allowed to correctly answer the math problems. Pressing START begins the game.

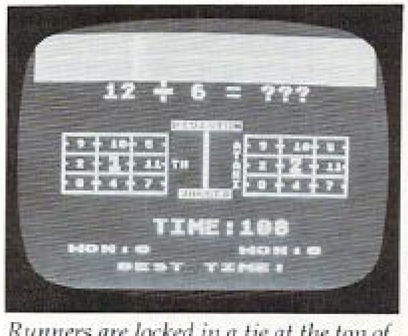
A math problem, nothing larger than two-digit numbers, will appear near the middle of the screen. Two matrices (one for each player) containing eight answers from which to choose appear on each side of the screen.

Once a player has located the correct answer on the matrix, he or she uses the joystick to position the cursor over the appropriate cell and presses the red button to indicate the answer. The player who gets the right answer first is rewarded with the advancement of his or her marathon runner at the top of the screen.

The game continues in this manner until one of the player's marathon runners crosses the finish line. On every fifth problem, the players are asked to identify the multiple of a given number. At the game's conclusion, the winning player will be ranked from "Walker-Team Six" to "Olympian-Team One." These rankings are derived from a combination of correct answers, advancements due to the opponent's incorrect responses, and the number of times that the timer expired.

Good Graphics

Geoff Brown, the author of *Marathon*, employs effective mixed screen modes and color schemes. The flow of the program is smooth and bug-free.



Runners are locked in a tie at the top of the screen while the program awaits the answer to a math problem in Marathon.

CENTURY MICRO PRODUCTS

****ALL SOFTWARE PRICES 25%-40% OFF RETAIL****

THE PRODUCTS BELOW ARE JUST AN EXAMPLE OF OUR SUPER DISCOUNTS. MANY MORE PRODUCTS ARE AVAILABLE AT SIMILAR SAVINGS FOR APPLE, IBM P/C, ATARI, TI PROFESSIONAL, COMMODORE, & TRS-80.

WRITE OR CALL FOR FREE CATALOG

APPLE	RETAIL	OUR PRICE
ASHTON-TATE		
dBase II (CP/M)	700.00	449.00
BRODERBUND		
Choplifter	34.95	25.00
Serpentine	34.95	25.00
Arcade Machine	54.95	40.00
CONTINENTAL		
The Home Accountant	74.95	49.95
1st Class Mail/Form Letter	99.95	64.95
EDU-WARE		
Compu-Read	29.95	21.95
Spelling Bee w/Read. Prim.	39.95	30.00
Compu-Math/Arith. Skills	49.95	35.00
INFOCOM		
Zork I, II, or III	39.95	27.95
Deadline	49.95	34.95
MICROSOFT		
Multiplan	275.00	189.00
MUSE		
Castle Wolfenstein	29.95	21.95
ON-LINE		
ScreenWriter II	129.95	85.00
General Manager	229.95	161.00
Frogger	34.95	24.00
Ultima II	59.95	42.00
SORCIM		
SuperCalc	295.00	180.00
SOFTWARE PUBLISHING CORP		
PFS:File (II or IIe)	125.00	88.00
PFS:Report (II or IIe)	125.00	88.00
PFS:Graph (II or IIe)	125.00	88.00
PFS:File/Report/Graph	375.00	239.95
SPINNAKER		
Snooper Troops I or II	44.95	32.00
Story Machine	34.95	25.00
Face Maker	34.95	25.00
STONEWARE		
D.B. Master	229.00	155.00
XEROX		
Sticky Bear Bop	39.95	29.25
Sticky Bear Numbers	39.95	29.25
Sticky Bear ABC	39.95	29.25
Old Ironsides	39.95	29.25
APPLE HARDWARE		
Kraft Joystick	64.95	49.00
Kensington System Saver Fan	89.95	69.00
Hayes Micromodem II w/Term. Prog.	409.00	299.00
Amdek Color I Monitor	399.00	319.00
Amdek Color I Plus (NEW!)	429.00	339.95
Amdek Color II Monitor	799.00	639.95
ATARI		
ATARI INC		
Centipede	44.95	32.00
Defender	44.95	32.00
E.T. Phone Home	49.95	37.50
Galaxian	44.95	32.00
Home Filing Manager	49.95	36.00
My First Alphabet	34.95	26.00
PILOT (Home Package)	79.95	59.00
16K RAM Memory Module	99.95	79.95
Word Processor	149.95	112.50
BRODERBUND		
Choplifter	34.95	25.00
Serpentine	34.95	25.00
CONTINENTAL		
The Home Accountant	74.95	49.95
DATASOFT		
Canyon Climber	29.95	21.95
Spell Wizard	79.95	58.50
EDU-WARE		
Compu-Read	29.95	21.95
Compu-Math/Fractions	39.95	30.00
Compu-Math/Decimals	39.95	30.00
INFOCOM		
Deadline	49.95	34.00
Zork I, II or III	39.95	27.00
ON-LINE		
Frogger	34.95	23.00
Ultima II	59.95	39.00
SPINNAKER		
Snooper Troops I or II	44.95	32.00
Story Machine	34.95	25.00
Face Maker	34.95	25.00
WICO		
Famous Red Ball Joystick	34.95	27.50
Trackball	69.95	52.00

TO ORDER: CALL 1-714-951-5596

8:00 A.M.-6:00 P.M. PST Mon.-Sat.
Orders will be credited for call

CENTURY MICRO PRODUCTS
P.O. Box 2520
Mission Viejo, CA 92690

Money Orders/Cashiers Checks accepted. Visa/Mastercard add 3%. Personal checks allow 2 weeks to clear. CA residents add sales tax. Shipping and handling add $3.00 (hardware extra). Prices subject to change.

Turning Logo Upside Down

Fred D'Ignazio, Associate Editor

In my February 1983 **COMPUTE!** column I put out a call for new computer languages for kids. I related my experiences with my own children (ages three and seven), trying to teach them to program. Frankly, I admitted that my efforts had failed. Neither child seemed to have the least inclination to learn how to program.

You readers responded to the column in a big way. You told me about your own thoughts about kids and programming, and you passed on news about programming projects and languages that you had learned about. I want to thank you for all this information. I will be printing excerpts from your letters in the coming months. Also, I am anxious to hear from even more of you. I believe kids' computer languages are the cutting edge of the revolution in computer learning. They deserve all the attention and debate we can muster up.

Enter *Delta Drawing*

Shortly after I wrote my February column, I received a copy of Spinnaker Software Company's *Delta Drawing* program for the Apple II Plus. (Versions for other computers are expected soon.) At first, I thought *Delta Drawing* was just another new "paint" program or simplified "turtle graphics" program. Then I dug further.

Now, my family and I have spent a dozen hours playing with *Delta Drawing*, and I am convinced that it is something more. I now believe that it is a first, but significant, step toward a powerful new computer language for kids.

I Want To Play Mystery House!

This is how my seven-year-old daughter, Catie, and I were first introduced to *Delta Drawing*:

I took the plastic off the *Delta Drawing* case and handed the disk to Catie. Catie put the disk in the Apple computer's disk drive and booted up the program.

We answered a couple of quick questions (did we have a color monitor? ‹Yes – a must›; did we have a printer? ‹No›). Then a little upside-down "V" appeared on the center of the screen. Under the V was a blinking dot. Around the edge of the screen was a blue box.

The manual calls the upside-down V the "Delta Drawing cursor." But Catie knew better. "That's a turtle," she said. "The blinking dot is her tail." Catie named the turtle DeeDee (for "DD" – Delta Drawing).

I thought things were going pretty well. That's when Catie got bored. (Catie gets bored easily.)

"I'm tired of this game," she said.

"But we haven't even started," I replied.

"I don't care. I want to play Mystery House."

Mystery House (from On-Line Systems) is one of Catie's favorite adventure games (along with *Cranston Manor*, also from On-Line, and *Copts and Robbers*, from Sirius).

Did I respond to Catie's obstinance with tact and gentle persuasion? Of course not. I did what any normal parent would do. I yelled at her.

Naturally, she responded by crying and I felt guilty. You really botched it, Fred, I thought to myself.

Fred D'Ignazio is a computer enthusiast and author of several books on computers for young people. His books include Katie and the Computer *(Creative Computing),* Chip Mitchell: The Case of the Stolen Computer Brains *(Dutton/Lodestar),* The Star Wars Question and Answer Book About Computers *(Random House), and* How To Get Intimate With Your Computer (A 10-Step Plan To Conquer Computer Anxiety) *(McGraw-Hill).*

As the father of two young children, Fred has become concerned with introducing the computer to children as a wonderful tool rather than as a forbidding electronic device. His column appears monthly in **COMPUTE!**.

Catie started banging on the Apple keyboard. All of a sudden, DeeDee came to life and drew a small straight line toward the top of the screen. She made a cute little "blink" or "clink" sound. Catie stopped crying and stared.

"Can I make my own pictures?" she asked.

Encouraged, I pulled out the "Fast Start" cards that accompany the *Delta Drawing* manual. Catie and I both avoided the manual. At 80 pages of fairly tiny print, it looked too intimidating, both for daughter and daddy.

But the Fast Start cards were different. Each one is made of shiny, plastic-coated, heavy-stock paper. Each one has a picture at the top and a few commands to show you how the picture was made. The cards are two-sided, numbered (with big numbers!) from one to fourteen.

One card tells you on one side how to load *Delta Drawing*, and on the back gives you all of the *Delta Drawing* commands. I discovered that Catie had pressed the Apple's "D" key and caused DeeDee to *draw*.

Catie and I looked at the pictures on the Fast Start cards. We grew excited. It looked like we could get DeeDee to draw all the standard stuff: circles, squares, triangles, and the like. We could also get DeeDee to draw three-dimensional cubes and "paint" the sides; play a game of Tic-Tac-Toe, and draw a colorful scene with an orange house, green grass, a blue sky, and an orange sun.

But how?

I did the adult thing and turned to card #1 and began trying to decipher the command beneath the picture. Meanwhile, Catie did the kid thing and began punching buttons. A couple of minutes later, I was still on card #1, but Catie had discovered that the "R" button made DeeDee turn right 30 degrees and the "L" button made her turn left 30 degrees. Catie showed me how she could press the "U" button and make DeeDee do a fancy U turn; and the "M" button to get DeeDee to scoot across the screen with her pen up – that is, she moved without drawing.

Catie squealed. "DeeDee didn't obey me," she said. "She went up and I wanted her to go down." We looked at the card with the command summary. We discovered that by pressing the "E" button we could make DeeDee backtrack and erase her last step.

I threw the Fast Start cards on the table. From that point on, we began improvising. Occasionally, we picked up the cards and borrowed commands off them, when we needed to make DeeDee do something we wanted her to do.

It sounds really impressive when I say that "Catie and I improvised." Actually, I advised Catie what to do, and she ignored me. This seemed to be a very successful strategy to learn *Delta Drawing*.

After a while, I gave up and let Catie take the lead.

Catie's approach was to do things with commands she already knew. For example, the first thing she did was hit the D (Draw) key until DeeDee drew herself off the top of the screen and popped back onto the bottom of the screen.

Catie wondered why DeeDee could "tunnel" off the screen, like Ms. Pac Man. Why didn't DeeDee bump her nose on the blue wall at the edge of the screen?

We looked at the Fast Start card with the command summary and found that there is a "B" command that makes DeeDee "bounce" instead of tunnel. When we pressed the B key, the wall turned green. To get DeeDee back into tunnelling mode, we had to press a "W" (wraparound) key.

Watching Catie at work was like watching a baby learn to speak for the first time – only in fast motion. She was learning a new language, and the moment she learned a new word in the language, she used it to express herself.

In fact, the reason she learned new words was to be able to express herself. She was motivated to master the language's vocabulary so she could do what she set out to do. And when she learned each new word, she automatically incorporated it into all the other words she knew. She was associating each word and developing word sequences – her own personal "grammar" in the new language.

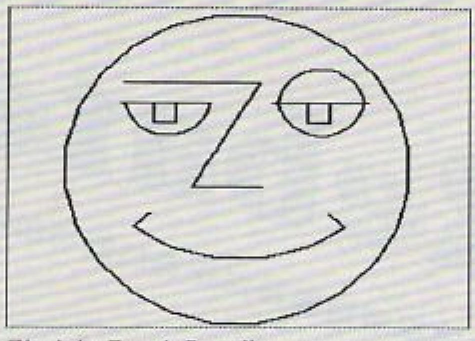
Catie's experimental approach was efficient, but it also led us occasionally into dead ends and surprises. For example, Catie got DeeDee to draw a house out of a square and a triangle. She filled the house with purple, by pressing "C" to choose the color, then by holding the CTRL button down and typing "F" (Fill). Then she tried to color the grass green. She pressed "C" and picked green. She typed CTRL-F, and green started washing like a wave across the screen. Then the green went out of control. It slipped through a tiny hole in the line that separated the grass and the sky, and it filled the sky, too. It ended up filling the entire picture, except for the purple house.

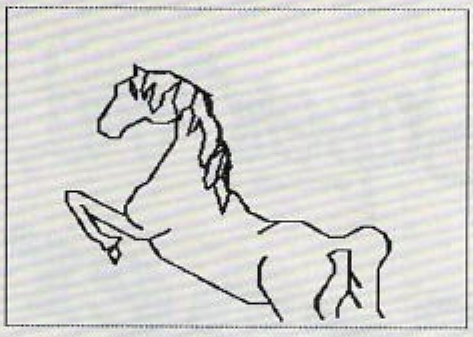
Catie howled!

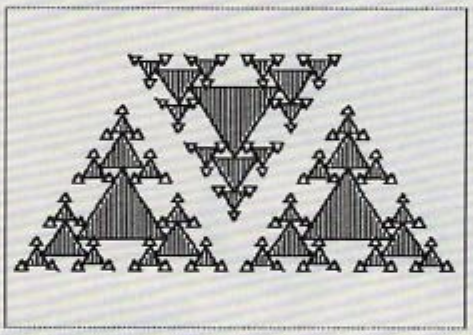
A Sun Not A Circle

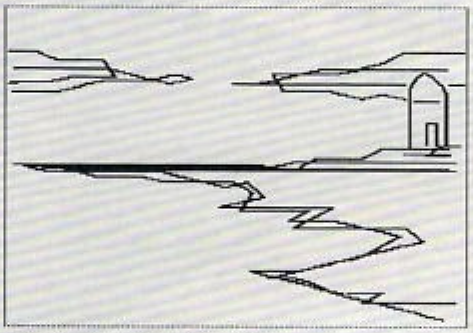
One thing that I immediately liked about *Delta Drawing* was the quick way Catie could make a picture. Also, I liked the precise, geometric way she constructed pictures. I have a problem with "paint" programs that use joysticks because my fine motor skills never passed the "klutz" stage. But, looking at Catie create pictures in *Delta Drawing*, I had hopes that even I might be able to make something pretty.

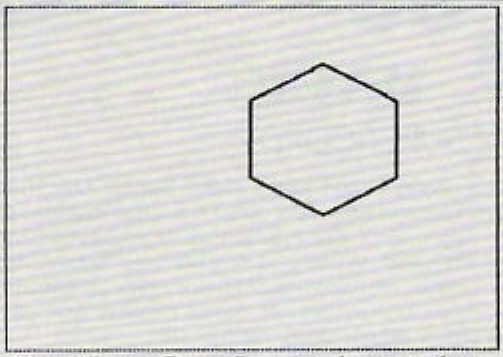
That is, if Catie would ever give me a turn.


Ziggie by Dennis Purcell

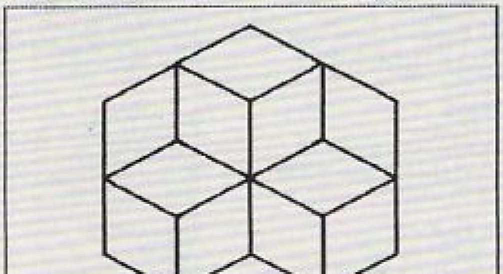

Pegasus by 12 year old computer summer camper


Triangle experiment by Jock Gill


Seascape by Clifford Wong



```
1<...  3D ..  >D ..  2R .....>1   hexagon element
2<...  6<1>..                     hexagon
```



```
1<...  3D ..  <D ..  2R   .....>1
2<...  6<1>..  2R   .....>2
3<...  6<2>..            hexagon with 60 degree turn
                         6 hexagons (motif #1)
```

Another thing that pleased me about *Delta Drawing* is that when children are using shapes like triangles, squares, and circles – they are not dealing with them in an abstract, adult sense. After all, shapes by themselves are boring. But shapes that resemble real-world objects like hats, planets, boxes, mountains, etc., are interesting. And shapes that can be combined into "building blocks" to make a new world are even more interesting. With *Delta Drawing*, Catie not only combined the shapes, she created new shapes to act as the proper building blocks for the world she was trying to create.

The Catie Robot Makes A Circle

With both Catie and Eric (three years old), I had tried the classic Logo experiment where you get the child to play "turtle" and figure out how to walk in a circle around the floor. I had often tried, but I had always failed.

The easy part was getting Catie and Eric to play turtle and figure out how to draw a circle. They took a baby step forward then made a small turn to the right. Then they repeated two steps over and over until they made a circle.

Fine. But then came the hard part – entering a program into the computer to make the turtle do what Catie and Eric had discovered so easily.

This is where I hit a brick wall. Catie and Eric had no interest in creating a "circle" procedure (program) in Logo. In fact, they never got past the first command – FORWARD or FD.

For Catie and Eric, it was too much effort for too little reward. And they didn't want to wait for the computer to learn the procedure. Why couldn't the computer obey them and make the circle immediately?

With Logo it couldn't, but with *Delta Drawing* it could. To make a circle, Catie typed D (Draw) and R (Right), then D and R, then D and R again. As she typed, DeeDee responded and drew the circle. It was easy to type D and R, and Catie got immediate results. After drawing the circle for the first time, she colored it orange with just a single command – CTRL-F. Then, on her own, she figured out how to make DeeDee move around the edge of the circle and make rays. She had turned her circle into a sun.

Our First *Delta Drawing* Program

Catie and I were doing great – until I accidentally bumped the "1" button on the keyboard and Catie's beautiful picture disappeared.

I thought she was going to kill me.

Quickly, I scanned the card with the command summary, looking for an "Unerase" button. "Why did that happen?" I grumbled. "Stupid program!"

I didn't find an unerase command on the card, but I did discover the "T" (text) command. I pushed the T button. DeeDee vanished. The screen filled with words.

There was Catie's program! It was still there.

I flipped through the big manual. A moment later I realized what I had done. By accident, when I pressed the "1" button, I had saved Catie's picture as a *program* – program #1. To get the picture back, I had to call the program. Doing that was unbelievably easy. I just had to push the "1" button again.

We switched back to DeeDee by pressing the "G" (Graphics) button. Then Catie pushed the "1" button, and, superfast, DeeDee drew and painted her picture. It was good as new.

Automatic Pictures

That's when Catie and I discovered the "A" button. The A button executes the Automatic command. The Automatic command automatically calls the last saved program and obeys it – over and over until you punch the ESC (escape) button.

I had a brainstorm. I had Catie type in a CTRL-D (a half-draw) and a CTRL-R (a half-turn right). DeeDee did her stuff.

Then I told Catie to press the "1" button. She did. Now we had a program that, when we pressed "T", looked like this:

```
1‹ ... ˆD .. ˆR ... ›1
```

The program looked puny and not very exciting. What was it good for? To find out, I asked Catie how we could make DeeDee automatically obey program #1 – over and over. With just a moment's thought, Catie pressed the "A" button.

A couple of seconds later, we had a circle!

To get DeeDee to stop drawing, Catie pressed the ESC button. Catie and I were excited: using program #1 as a building block, we had created a "circle" program – program #2. We saved program #2 by pressing the "2" button.

We drew circles all over the picture screen and got DeeDee to paint them different colors. Then we tried something simpler – and neater.

We erased all our current commands by typing CTRL-E. We pressed the "T" button. Our first two programs were still intact.

We pressed 2 and got a quick circle. Then we pressed the "L" button. DeeDee turned 30 degrees to the left. We saved these two commands as program #3. The first three programs looked like this:

```
Program #1   1‹ ... ˆD .. ˆR ... ›1
Program #2   2‹ ... 25‹1› ... ›2
   This came from pressing the A button.
Program #3   3‹ ... ‹2› .. L ... ›3
```

When you see a number inside brackets, like ‹2›, it means you are calling a program – this time program #2. The 25‹1› means you are calling program #1 25 times. We got the computer to do this just by pressing the A button – once! – and the ESC button to stop DeeDee.

Next Catie typed the A button. DeeDee drew a circle, turned left 30 degrees, drew another circle, turned left 30 degrees, drew another circle, and so on. In about a minute she had rotated her way around the picture screen. She had drawn a three-dimensional figure: a doughnut! Catie, on her own, moved DeeDee and had her paint the doughnut's center orange and the background violet. *After the doughnut was drawn* (not before), Catie and I pressed "T" to see what the program looked like. Here it is:

```
Program #4   4‹ ... 44‹3› .. 2L .. M .. C:2
                 .. ˆF .. 8M .. C:5 .. ˆF ..
```

Remember: the most formidable command – 44 "calls" of program #3 was achieved by pressing the A button *once*.

Catie and I saved our doughnut in just a few seconds. We pressed CTRL-S, and the computer asked if we wanted to save or recall (load) something. We pressed "S" for save. Then the computer asked us if we wanted to save the program ("T" – text) or the picture ("G" – graphics). We typed "T." The computer told us to load in our own disk. We did, and it asked us to name our *Delta Drawing* file – we called it DONUT. Then the computer saved it.

Nested Building Blocks

Delta Drawing's real power comes from its ability to save pictures as building blocks; from its ability to combine simple building blocks into blocks that are more and more elaborate and complex. And you can gain access to all these building blocks just by pressing the CTRL-A buttons. When you press CTRL-A, the computer asks you which building block (program) you want. You can choose any number, from 1 to 9.

By pushing just a few buttons, Catie and I created our doughnut. We built the doughnut from a draw and turn program, a circle program, and a circle and turn program. Just as easily, we could have created "house" programs, "people" programs, "tree" programs, and so on. We could have formed a picture by positioning DeeDee on the screen and calling the program we wanted.

A Kid's Language

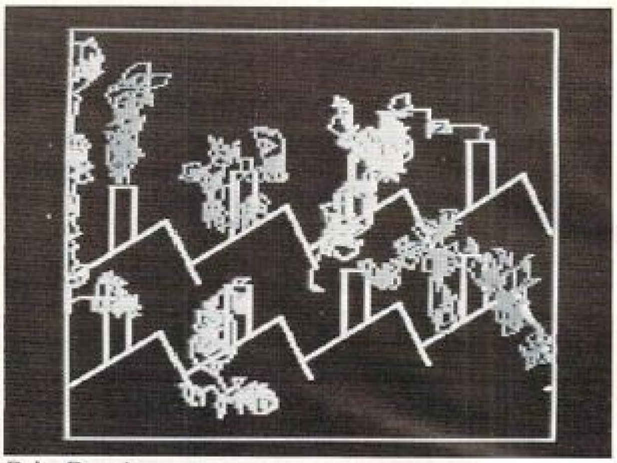
Delta Drawing represents, I hope, one of the early representatives of a new generation of children's software that combines simplicity with great power. Also, it is open-ended. It is a language. Once the child learns the language, she can do whatever she wants. And she can do a lot *even as she is learning the language*.

This simplicity, power, and freedom are what made the program a hit with Catie. And when her brother showed up, and learned a few buttons, they made a big hit with him, too. All of a sudden programming becomes an activity with immediate results that are meaningful to the child, controlled by the child, and that challenge and stimulate the child to be original and inventive.

This is certainly a good start toward a kids' language of the future. And it's not a bad adult's language, either. My wife Janet and I have had a ball creating pictures with *Delta Drawing*. Here is a piece of software that is equally fascinating and easy to use for a three-year-old, a first grader, and two jaded adults. The generation gap between the different members of the family disappears when we use *Delta Drawing*. We are all equally caught up in exploring its possibilities. And no single member of the family seems to have an edge. This is a very nice feature of the program.

Upside Down Logo

Why did I claim that *Delta Drawing* is like Logo turned on its head? Because with Logo (and most other languages), you have to type in the commands in your program before you can run the program and create a picture. With *Delta Drawing* you make the picture first, and in making the picture you create a program. It's just the opposite. It's Logo upside down!


Delta Drawing

Delta Drawing costs $59.95 and runs on the Apple II+. By spring, a new version will be available on the IBM PC. By next fall you can look for it on the Atari 800, the VIC, the Commodore 64, and possibly on other low-priced computers.

To inquire about *Delta Drawing*, write:

Spinnaker Software Corporation
215 First Street
Cambridge, MA 02142
617-868-4700

New Resources

Since my last column I've received two interesting new books.

The Computer Camp Book

The Computer Camp Book is published by The Yellow Springs Computer Camp, Inc. It has 224 pages and costs $12.95. To order the book, write:

The Computer Camp Book
1424 Glen View Drive
Yellow Springs, OH 45387

or call 513-767-7717.

The book is a wealth of information about computer camps, including:

- How to start and run your own computer camp
- Ideas and materials for teaching and learning
- Computer literacy activities
- A look at different computer camps
- A nationwide guide to computer camps, courses, and workshops
- A guide to computing resources (including a five-page guide to computing resources for handicapped people)

Parent's Guide to Computers in Education

The *Parent's Guide* was written by David Moursund. It is a real buy – 80 pages for only $3.50. To get the *Guide*, write:

Parent's Guide
Dept. of Computer & Information Science
University of Oregon
Eugene, OR 97403
or call 583-686-4429.

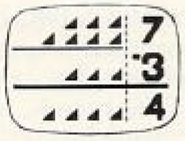
The book covers a lot of ground in a clear, simple style. Some of the subjects covered include: the school of the future, introduction to computers, hardware and software, computers in education, "What You Can Do," a buyer's plan, a glossary, and a list of resources.

A unique feature of the book is that it is really two books in one. A second book, entitled "Here Comes the Dawn (If Only I Can Find the Switch)," written by Merle Marsh, appears in little italicized text boxes at the foot of every few pages. It begins, on page 3: "I tried to enter the Computer Age by quietly sneaking up on the new technology ...."

©

C-64 TI99/4A TIMEX VIC 20 ATARI

CHILD DEVELOPMENT SERIES
(for the 3.5K VIC and 16K ATARI)



ADD/SUB—$16.95
Displays single or multiple digits with or w/o pictures, borrows, carries, scoring, and audio/video feedback.

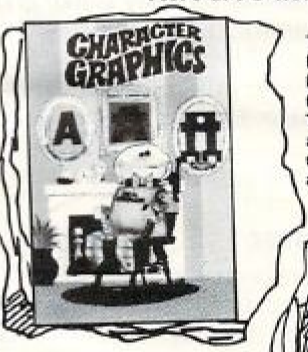
NUMER-BECi—$16.95
Number recognition, object counting, object grouping, and number/size/shape discrimination.

BECi is composed of professionals dedicated to providing non-trivial educational materials for the home computer. In addition to our own software, we carry a full line of evaluated hardware and software. Send $2 (refundable) for our catalog.

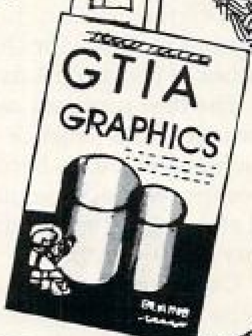
Send check or money order to:
BOSTON EDUCATIONAL COMPUTING, INC.
78 Dartmouth Street, Boston, MA 02116
(617) 536-5116 *MA res. add 5% tax

See Us At THE EAST COAST COMPUTER FAIR

DIGGING AROUND FOR SOME NEW TUTORIALS?
Here are 3 new ones from EDUCATIONAL SOFTWARE inc:



TRICKY TUTORIAL #8
Learn to take the shapes of the letters and numbers on the keyboard and change them to anything you want. Special fonts and creatures that move are all explained in this huge Tutorial. We'll tell you how a shape is stored in memory, animate several shapes, then give you a game using all the techniques. Includes an editor that animates your characters as a preview. 16K Tape / 32K Disk $29.95



TRICKY TUTORIAL #9
Learn to use Graphics modes 9, 10 and 11 that your BASIC Manual forgot to tell you about! There are 16 shades or 9 colors awaiting you within the machine. Prototype will take you through many examples while Professor Von Chip explains how they were done. Includes three dimensional shapes and digitized pictures. 16K Tape / 32K Disk $29.95

TRICKY TUTORIAL #10
Over the years, Professor Von Chip has acquired a large collection of sound effects on his ATARI that are unusual to say the least! We offer them here plus an ample amount of explanations, utilities and graphics to save you endless hours. Have fun while learning! 16K Tape / 32K Disk $29.95

AVAILABLE FROM MOST DEALERS
WRITE FOR A FREE CATALOG
4565 Cherryvale Ave., Soquel, CA 95073
CALL FOR ORDERING INFORMATION
MC/VISA/COD: (800) 692-9520 OR (408) 476-4901

EDUCATIONAL SOFTWARE • E.S.I. PROGRAM EXCHANGE • PROTO

Apple Shape Generator

J. F. Johnson

The Apple computer allows shapes to be manipulated from within a BASIC program. Although shapes are very useful in two-dimensional dynamic graphics, the process of creating shapes and entering them into a shape table is tedious, and errors are exceedingly difficult to correct. This program simplifies the process of defining a shape. All required binary to hexadecimal conversions require no user intervention and a shape table is automatically constructed, with each new shape added to the current table.

Many of the shape-drawing routines currently available for the Apple allow a shape to be created within a rectangular drawing area, with a bit map of this entire area, then stored as the shape. This technique is fine for creating relatively small shapes. However, as the size of the shape increases (so that the rectangular area the size of either hi-res page is required to enclose the shape), the amount of wasted space (i.e., bytes which are "off" and represent only the background) becomes considerable. A bit map of a shape requiring a rectangular area of this size would require 7-8 K.

This program creates shapes in the manner explained in the Applesoft manual (Chapter 9). The head-to-tail vector method is used to initially define the shape. These vectors are then "unwrapped" and sequentially combined in pairs for conversion from their individual binary codes into equivalent hexadecimal code. Each hexadecimal byte represents one byte in the shape definition. The shape is then added to the table in memory with the table's index also updated. Shapes which would have required up to 8K in a table have been reduced to less than 1K using this program.

Capabilities Of Key Shape Maker

The following can be accomplished with Key Shape Maker:

1) Construct a shape table comprised of 1-255 shapes.
2) Create a table with a maximum length of 6K.
3) Alter any shape after it has been entered into the table. Also, add "buffer bytes" at the end of each shape definition so that any shape can be slightly enlarged relative to its original definition.
4) Correct mistakes which occur while entering vectors during a shape definition by erasing them in a sequential fashion.
5) View all the shapes in the current table (using the game paddles).
6) Display any particular shape, with the effect of ROT and SCALE variations (using the game paddles) on the shape immediately displayed on the hi-res screen.
7) Once a shape table is BSAVEd to diskette using this utility, it may be BLOADed with the utility and the stored shapes redefined and new shapes added (assuming the table does not contain the maximum number of shapes originally designated).
8) The current shape table in RAM can be destroyed, and a new table created or an old table BLOADed into memory.

Use An EXEC File To Initialize

The entire program is written in Applesoft. The following program creates a text file, "Key Shape Loader", which reassigns the beginning of the program pointer (104, 103) and then RUNs the program.

Program 1.

```
5  REM  KEY SHAPE LOADER MAKER
10 D$ =  CHR$ (4)
15  PRINT D$"MON C,I,O"
17  PRINT D$"DELETE KEY SHAPE LOADER"
20  PRINT D$"OPEN KEY SHAPE LOADER"
30  PRINT D$"WRITE KEY SHAPE LOADER"
40  PRINT "POKE 104,96"
50  PRINT "POKE 103,1"
60  PRINT "POKE 24576,0"
70  PRINT "RUN KEY SHAPE MAKER"
80  PRINT D$"CLOSE KEY SHAPE LOADER"
90  END
```

By EXECing the text file Key Shape Loader, the required POKEs are completed, and then the Applesoft program "Key Shape Maker" is RUN.

Use Of RAM By "Key Shape Maker"

The Applesoft program is LOADed at $6001 (24577), just above the second hi-res page of graphics. The second hi-res page is used for the temporary storage of vectors that define the current shape. These vectors are then paired and converted into their equivalent hexadecimal code, with the resulting hex code defining the shape stored on the second hi-res page. If the shape is to be saved, the hex code is then transferred to the shape table. The creation and display of all shapes utilizes the first hi-res page. The shape table is stored at $800 (2048), and its length may not exceed $2000 (8196) since the first hi-res page is used for display purposes.

Execution

The user is initially prompted for the number of shapes that will be entered into the table. Since extra shapes are invariably required at a future date, it is always best to enter a number larger than what is currently estimated. The minimum number is 1, and the maximum is 255. Since the table need not be completed at one setting, the partially constructed table can be BSAVEd, then BLOADed at a future date, with additional shapes added (up to the original number that was user-specified) or current table shapes redefined.

This maximum number of shapes is then POKEd into $801. Room for the shape table index (which immediately follows starting at $802) is then allocated. The index stores the locations of all shapes relative to the start of the table ($800). The index must contain two bytes for each stored shape. If the estimated number of shapes to be stored in this table is low, it will not be possible to exceed this limit since room in the table for the index can not be changed using this program. Location $800 initially contains a value of zero, and is incremented by one upon the addition of each shape to the table.

The shapes are created using two different sets of four keys. Plotting vectors are entered using the I, K, M, and J keys, while the nonplotting vectors are entered using the E, D, X, and S keys. Both sets of keys are arranged on the keyboard in a north-east-south-west fashion, with the right-hand set for plotting and the left-hand set for nonplotting. The back arrow key (<-) may be used to sequentially erase vectors starting with the last one entered, and is very useful for correcting any mistakes. The keystroke "!" (a shift-1) terminates the shape definition.

Prior to the actual construction of the shape, a "dot-cursor" is positioned on the first hi-res screen. This is the point at which the shape definition is initiated. The shape is then displayed as it is constructed, using the previously defined keystrokes. Due to the algorithm used to display the shape as it is defined, any nonplotting vectors which cross any existing outline of the shape will result in the boundary being erased where the crossover occurs.

However, when the final shape is displayed for verification, it will exhibit the contiguous boundary that was originally constructed. Also displayed during the construction of the shape are the current x and y coordinates of the "dot-cursor," the three-digit binary code of each vector as it is entered, and the maximum number of bytes which may be used to define the present shape.

When the definition of the shape is terminated, the keystroke vectors are converted to hexadecimal code, with the resulting shape displayed prior to its storage in the table. If the user chooses to save the shape, he or she appends it to the current table, updates the corresponding index locations, and increments location $800 by one. If the shape is not saved, the defining of additional shapes simply continues.

Applesoft Shape Table Commands

Several subroutines in this program allow the user to experiment with several shape table commands and to view the result. This was purposely included to aid the user in exploring the capabilities (as well as the limitations) of shapes within Applesoft. This will perhaps facilitate inclusion of shape tables within programs.

The SCALE command allows the expansion of a defined shape. Since the originally defined shape is constructed using the smallest SCALE value, a figure may only be expanded using this command. It will soon be discovered, however, that the contiguous boundary of a shape may become segmented when its size is enlarged through SCALEing, and may rapidly become unrecognizable. This can usually be overcome by redefining the same shape boundary using a different sequence of plotting/nonplotting vectors. The ability to redefine any given shape will allow the user to experiment.

Rotations in the plane of the screen are controlled by the ROT command. An inverse relationship exists between the number of unique rotational values defined by the ROT command and the SCALE command. Increasing ROT from 0 to 64 will rotate it 360 degrees about the origin. As the value for SCALE increases from 0, more unique rotational values are recognized between the ROT values of 0 and 64, and hence the incremental rotational angle decreases. By making the original shape very small, and then expanding it using the SCALE command, a smaller angle of rotation can be realized between the ROT values of 0 and 64. The values for both of the commands may be varied for a chosen shape, with the effects on the shape displayed on the screen.

A shape may be displayed from Applesoft using either the DRAW or XDRAW commands. The XDRAW command simply complements the current color of the shape at its present location and is very convenient for displaying and erasing shapes. The DRAW command requires that HCOLOR be changed from a value of 3 to 0 if the shape is to first be drawn and then erased. These commands may also display the same shape differently. If any nonplotting vectors cross the boundary of plotting vectors in the original shape definition, the DRAW command (HCOLOR=3) will display a contiguous shape.

The XDRAW command, however, displays the shape with any regions of plotting/nonplotting vector overlap being effectively erased. This should be taken into consideration when originally defining the shape boundary, since one of the two display techniques may be preferred in the Applesoft program which uses the shapes. The shape display for verification purposes (prior to appending the shape to the current table) is displayed using DRAW (HCOLOR=3). During viewing of a shape in the current table with ROT and SCALE variations, the shape is drawn and erased using XDRAW.

Using A Shape Table

Key Shape Maker creates a shape table starting at $800 (2048) in RAM. It may be BLOADed into another region if there exists a conflict with the storage of the controlling Applesoft program or a machine language program which must occupy this region. There are two DOS entry points which store both the starting address and length of a BLOADed file. Since the user determines the starting address of a binary file, only the length must be determined. This is accomplished in the following manner.

After BSAVEing your shape table to diskette, BLOAD it back into memory (this may be done in direct execution or under Key Shape Maker control). If the shape table has been loaded by an Applesoft program, press the reset button. Now enter the following as a direct execution instruction, where ‹ret› simply designates pressing the return key.

```
PRINT PEEK(43616) + PEEK(43617) * 256 <ret>
```

The base ten number that appears on the screen immediately after this instruction is the length of the shape table (see Appendix E of the DOS manual, DOS Entry Points And Schematics). Using this additional piece of information, the user is offered some flexibility in BLOADing the shape table into various regions of RAM. For example, a shape table of byte length 350 may be BLOADed at location 24577 (immediately above the second hi-res page) with the following instruction in an Applesoft statement.

```
100 PRINT CHR$(4) "BLOAD SHAPE TABLE-1,
    A24577, L350"
```

The final piece of information which must be supplied is the location of the shape table. The pointer designating the beginning of the current shape table is located on the zero page of memory, and is comprised of the locations $E8 (232) and $E9 (233). The integer value obtained by dividing the starting address by 256 is POKEd into 233, with the remainder POKEd into 232 (i.e., 24577/256=96 with a remainder of 1).

```
110 POKE 233,96 : POKE 232,1
```

Your Applesoft program will now be able to effectively use the shape table currently residing in RAM.

Program 2.

```
5  REM  KEY SHAPE MAKER
60  REM    TS=START OF SHAPE TABLE///VC=MARK
    ER USED IN DISPLAY OF 6 DIGITS REPRESEN
    TING 2 VECTORS///VS=MARKER FOR START OF
     TEMPORARY STORAGE FOR VECTOR TABLE AND
     ENSUING TEMPORARY STORAGE DERIVED SHAP
    E///16395=START OF TEMPORARY SHAPE TABL
    E
70 A$ = "PRESS ! TO STOP DRAWING SHAPE."
80 TS = 2048: POKE TS,0:VC = 16389:VS = 1639
    6:LI = 2050:MI = 2051:D$ =  CHR$ (4): GOTO
    4000
100  HCOLOR= 3: HPLOT X,Y: FOR J = 1 TO 20: NEXT
    J: HCOLOR= 0: HPLOT X,Y:X =  PDL (0) /
    .913:Y =  PDL (1) / 1.6: IF  PEEK ( - 1
    6287) > 127 OR  PEEK ( - 16286) > 127 THEN
     RETURN
105  GOTO 100
110 S1 =  INT (1 +  PDL (0) * ( PEEK (TS) -
    1) / 240): ROT= 0: HCOLOR= 3: SCALE= 1:
     RETURN
115 S2 =  INT (1 +  PDL (0) * ( PEEK (TS) -
    1) / 240): RETURN
120  XDRAW S1 AT X,Y: VTAB 24: HTAB 1: CALL
     - 868: PRINT "SHAPE #"S1".";
125  GOSUB 115: IF  PEEK ( - 16287) > 127 THEN
     RETURN
130  IF S2 <  > S1 THEN  XDRAW S1 AT X,Y:S1 =
    S2: GOTO 120
135  GOTO 125
140  GOSUB 110
145  VTAB 5: HTAB 1: CALL  - 868: PRINT "SHA
    PE #"S1"."
150  GOSUB 115: IF S2 <  > S1 THEN S1 = S2: GOTO
    145
152  IF  PEEK ( - 16287) > 127 THEN  RETURN
154  GOTO 150
158 S1 =  INT ( PDL (1) * 7 / 240): RETURN
159 S2 =  INT ( PDL (1) * 7 / 240): RETURN
160  GOSUB 158
162  VTAB 10: HTAB 1: CALL  - 868: PRINT "HC
    OLOR="S1"."
164  GOSUB 159: IF S2 <  > S1 THEN S1 = S2: GOTO
    162
166  IF  PEEK ( - 16286) > 127 THEN  RETURN
168  GOTO 164
170  GOTO 166
172 R1 =  PDL (0) / 3:S1 =  PDL (1) / 3: RETURN
173 R2 =  PDL (0) / 3:S2 =  PDL (1) / 3: RETURN
174  GOSUB 172
175  HCOLOR= HC: ROT= R1: SCALE= S1: DRAW SH
     AT XI,YI: VTAB 24: HTAB 1: CALL  - 868
    : PRINT "ROT=" INT (R1) SPC( 8)"SCALE="
     INT (S1);
176  GOSUB 173: IF R2 <  > R1 OR S2 <  > S1 THEN
    R1 = R2:S1 = S2: CALL 62450: GOTO 175
177  IF  PEEK ( - 16287) > 127 OR  PEEK ( -
    16286) > 127 THEN  RETURN
178  GOTO 176
200  POKE TS + 1, VAL (NS$): RETURN : REM  M
    AXIMUM NUMBER OF SHAPES THAT CAN BE ENT
    ERED INTO THIS TABLE
203 PA = 256 *  PEEK (MI) +  PEEK (LI) + TS:
     RETURN
205 PA = TS + 4 + 2 *  VAL (NS$): RETURN : REM
       IS LOCATION IN TABLE WHERE FIRST SHA
    PE WILL BE SAVED
210 LS = TS + 2 * SH:MS = TS + 1 + 2 * SH:DD
     = 256 * ( PEEK (MS + 2) -  PEEK (MS)) +
    ( PEEK (LS + 2) -  PEEK (LS)): RETURN
```

```
212 LI = TS + 2 * ( PEEK (TS) + 1):MI = LI +
     1: RETURN : REM   INIT INDEX FOR TABLE
     THAT HAS BEEN LOADED
215 LI = LI + 2:MI = MI + 2: RETURN : REM
     INCREMENT INDEX LOCATION FOR NEXT SHAPE

220 LI = LI - 2:MI = MI - 2: RETURN : REM
     DECREMENT INDEX LOCATION FOR FIRST SHAP
     E TO BE DRAWN IN LOADED OR ALTERED TABL
     E
225 IP = VS: RETURN : REM  INITIALIZE LOCATI
     ON WHERE PLOTTED VECTORS ARE STORED TEM
     PORARILY UNTIL THEY ARE CONVERTED INTO
     A SHAPE
230 N = VS + 1:SL = VS + 1: RETURN : REM  IN
     ITIALIZE TWO COUNTERS WHICH ARE USED DU
     RING THE CONVERSION OF STORED VECTORS I
     NTO A SHAPE
235 PA = TS + 256 *  PEEK (MS) +  PEEK (LS):
      RETURN : REM  LOCATION IN TABLE OF STA
     RT OF NEXT SHAPE
240  POKE LI, INT ((((PA - TS) / 256) -  INT
     ((PA - TS) / 256)) * 256 + .5): POKE MI
     , INT ((PA - TS) / 256): RETURN : REM
       POKE STARTING LOCATION FOR GIVEN SHAP
     E IN APPROPRIATE INDEX LOCATION
250 A = 0:B = 0:C = 0: RETURN : REM   INITIA
     LIZE A,B,C TO ZERO
255 L = IP - VS:K =  INT (L / 2) +  INT ((L /
     2 -  INT (L / 2)) * 2 + .05): RETURN : REM
      L=#BYTES CONTAINING VECTORS///K=#BYTES
      REQUIRED TO STORE SHAPE;1 SHAPE BYTE P
     ER 2 VECTOR BYTES
260  POKE 233,64: POKE 232,9: POKE 16393,1: POKE
     16395,4: POKE 16396,0: RETURN : REM   D
     EFINE UNIT SHAPE TABLE WHERE TEMPORARIL
     Y DEFINED SHAPE EXISTS
265  POKE 233,8: POKE 232,0: RETURN : REM  L
     OCATION OF SHAPE TABLE
270 RS =  PEEK (TS + 1) -  PEEK (TS): RETURN
     : REM   RS=# OF SHAPES THAT MAY STILL B
     E ENTERED INTO SHAPE TABLE
299  REM  PLOT/ERASE POINT AT CURRENT X,Y UN
     TIL KEY PRESS OCCURS.
300 XO = X:YO = Y: HCOLOR= 3: HPLOT XO,YO: FOR
     J = 1 TO 20: NEXT J: HCOLOR= 0: HPLOT X
     O,YO: FOR J = 1 TO 20: NEXT J: IF  PEEK
     ( - 16384) < 128 THEN 300
310  HCOLOR= 3: POKE  - 16368,0:Z =  PEEK ( -
     16384): RETURN
324  REM   PLOT PRESENT POINT IF ENTERED VEC
     TOR IS A PLOT-THEN-MOVE VECTOR
325  HCOLOR= 3: HPLOT XO,YO: RETURN
329  REM  ERASE PREVIOUS POINT PLOTTED
330  HCOLOR= 0: HPLOT XO,YO: RETURN
349  REM  EVALUATE KEY PRESS IN TERMS OF NEW
     X,Y COORDINATES.
350 F1 = 0
352  IF Z = 73 OR Z = 69 THEN Y = Y - 1: GOSUB
     362: RETURN : REM  MOVE UP
354  IF Z = 75 OR Z = 68 THEN X = X + 1: GOSUB
     364: RETURN : REM   MOVE RIGHT
356  IF Z = 77 OR Z = 88 THEN Y = Y + 1: GOSUB
     366: RETURN : REM  MOVE DOWN
358  IF Z = 74 OR Z = 83 THEN X = X - 1: GOSUB
     368: RETURN : REM   MOVE LEFT
360 F1 = 1: RETURN : REM   FLAG F1 SET TRUE
     IF NO U,R,D,L MOVE
362  IF Y < 0 THEN Y = 0:F1 = 1
363  RETURN
364  IF X > 279 THEN X = 279:F1 = 1
365  RETURN
366  IF Y > 159 THEN Y = 159:F1 = 1
367  RETURN
368  IF X < 0 THEN X = 0:F1 = 1
369  RETURN
399  REM  EVALUATE 3 DIGIT BINARY EQUIVALENT
      OF INDIVIDUAL VECTOR
400 F1 = 0: IF Z = 73 THEN A = 1:B = 0:C = 0
     : RETURN
402  IF Z = 75 THEN A = 1:B = 0:C = 1: RETURN

404  IF Z = 77 THEN A = 1:B = 1:C = 0: RETURN

406  IF Z = 74 THEN A = 1:B = 1:C = 1: RETURN

408  IF Z = 69 THEN A = 0:B = 0:C = 0: RETURN

410  IF Z = 68 THEN A = 0:B = 0:C = 1: RETURN

412  IF Z = 88 THEN A = 0:B = 1:C = 0: RETURN

414  IF Z = 83 THEN A = 0:B = 1:C = 1: RETURN

418 F1 = 1: RETURN
424  REM  PRINT PRESENT COORDINATES OF X,Y
425  VTAB 21: HTAB 1: CALL  - 868: PRINT "X=
     "X,"Y="Y: RETURN
449  REM  ERASE CURRENT POINT AND MOVE BACK
     ONE POINT
450 PP =  PEEK (IP): IF IP = VS THEN  RETURN
     : REM  CAN'T ERASE PAST ORIGIN OF SHAPE

455  IF PP = 0 OR PP = 4 THEN Y = Y + 1: GOSUB
     475: RETURN
460  IF PP = 1 OR PP = 5 THEN X = X - 1: GOSUB
     475: RETURN
465  IF PP = 2 OR PP = 6 THEN Y = Y - 1: GOSUB
     475: RETURN
470  IF PP = 3 OR PP = 7 THEN X = X + 1: GOSUB
     475: RETURN
475 XO = X:YO = Y: GOSUB 330: POKE IP,0:IP =
     IP - 1: RETURN
499  REM  POKE VECTOR INTO RAM LOCATION IP
500 IP = IP + 1: POKE IP,4 * A + 2 * B + C: RETURN
509  REM  POKE BINARY EQUIVALENT OF VECTOR M
     OVE
510 P(1 + I * 3) = A:P(2 + I * 3) = B:P(3 +
     I * 3) = C
515  IF I = 1 THEN  FOR J = 0 TO 5: POKE 187
     2 + J,48: NEXT J: FOR J = 0 TO 2: POKE
     1875 + J,P(4 + J) + 48: NEXT J: RETURN

520  FOR J = 0 TO 2: POKE 1872 + J,P(1 + J) +
     48: NEXT J: RETURN
525  FOR J = 1 TO 6:P(J) = 0: NEXT J: RETURN

600  HGR2 : HGR : SCALE= 1: ROT= 0: HCOLOR=
     3:XX = 139:YY = 80:X = XX:Y = YY: RETURN
     : REM    HI-RES INITIALIZATION
700 BL = 8190 - PA:DI = 24576 - 16396:VL = D
     I: RETURN : REM  NEW TABLE BYTE LIMITS
710 NS =  PEEK (2048):LI = TS + 2 * (NS + 1)
     :MI = TS + 1 + 2 * (NS + 1):PA = TS + 2
     56 *  PEEK (MI) +  PEEK (LI)
720 BL = 8190 - PA: IF DI < 2 * (8190 - PA) THEN
     VL = DI: RETURN
730 VL = 2 * (8190 - PA): RETURN
765 F1 - 0: IF VL < 100 THEN F1 = 1
767  RETURN
770 F2 = 0: VTAB 21: PRINT "THERE ARE "8190 -
     PA" BYTES REMAINING FOR MORE": PRINT "S
     HAPES IN CURRENT TABLE IF YOU HAVE NOT
     CONSTRUCTED THE LAST SHAPE."
775  IF 8190 - PA < 100 THEN  PRINT "NO MORE
      SHAPES MAY BE ADDED TO CURRENT  TABLE.
     ":F2 = 1
780  RETURN
800 F3 = 0:VL = VL - 1: VTAB 21: HTAB 33: CALL
      - 868: PRINT VL
805  IF VL < 200 THEN  VTAB 22: HTAB 1: PRINT
     "ONLY "VL - 190" MOVES LEFT.";: FOR J =
     1 TO 1000: NEXT J: HTAB 1: CALL  - 868:
      IF VL <  = 191 THEN F3 = 1
810  RETURN
975  VTAB 24: HTAB 5: CALL  - 958: PRINT "PR
     ESS ANY LETTER TO CONTINUE.";: GET Z$:J
      =  FRE (0): RETURN
999  REM  INITIALIZE SHAPE TABLE PARAMETERS
1000  TEXT : HOME : PRINT  TAB( 5);"THE NUMB
     ER OF SHAPES THAT MAY BE    ENTERED IN
     A SHAPE TABLE IS IN THE RANGEOF 1-255.
      IT IS ALWAYS BEST TO ALLOW   EXTRA ROO
     M FOR ADDITIONAL SHAPES YOU MAYWISH TO
     INCLUDE IN THE FUTURE."
1010  INPUT "      ENTER A NUMBER BETWEEN 1 A
     ND 255,  THEN PRESS RETURN.";NS$: IF  VAL
     (NS$) < 1 OR  VAL (NS$) > 255 THEN 1000

1020  GOSUB 200: REM   POKE MAX # OF SHAPES
     THAT CAN BE ENTERED INTO THIS TABLE
1030  GOSUB 205: REM  INITIAL RAM LOCATION F
     OR FIRST SHAPE
1040  GOSUB 240: REM  STARTING POINT OF FIRS
```

```
     T SHAPE AND POKE INTO INDEX
1050  GOSUB 700: REM  BL,DI,VL
1055  RETURN
1060  GOSUB 250: REM  INITIALIZE COMPONENTS
     OF VECTOR MOVE
1070  GOSUB 600: REM  HI-RES INIT
1080  RETURN
1200  TEXT : HOME : PRINT  TAB( 5);"BEFORE A
     CTUALLY DRAWING A SHAPE,     THE BLINKIN
     G DOT MAY BE MOVED TO ANY     POSITION O
     N THE SCREEN.  USE THE E,S,D  AND X KEY
     S FOR DOT POSITIONING ONLY."
1205  PRINT "PRESS ! WHEN READY TO DRAW A SH
     APE."
1210  PRINT  TAB( 5);"THE SHAPE YOU ARE TO D
     RAW MAY THEN  BE COMPRISED OF PLOTTING
     AS WELL AS NON PLOTTING VECORS.  USE TH
     E E,S,D AND X   KEYS FOR NONPLOTTING VE
     CTORS, AND THE   I,J,K AND M KEYS FOR P
     LOTTING VECTORS."
1220  PRINT "THE LEFT ARROW KEY (<-) MAY BE
     USED TO  ERASE MISTAKES, AND THE ! KEY
     TO TERM-  INATE THE SHAPE.": PRINT
1240  PRINT  TAB( 1);"-NONPLOTTING-"; SPC( 9
     );"-PLOTTING-"
1250  PRINT  TAB( 2);"E-MOVE UP" SPC( 7)"I-P
     LOT THEN MOVE UP": PRINT  TAB( 2);"S-MO
     VE LEFT" SPC( 5)"J-PLOT THEN MOVE LEFT"
1260  PRINT  TAB( 2);"D-MOVE RIGHT" SPC( 4)"
     K-PLOT THEN MOVE RIGHT": PRINT  TAB( 2)
     ;"X-MOVE DOWN" SPC( 5)"M-PLOT THEN MOVE
      DOWN"
1270  PRINT  TAB( 12);"<-  ERASER": PRINT  TAB(
     12);"!   STOP"
1280  GOSUB 975
1285  HOME : VTAB 23: PRINT "PRESS ! WHEN YO
     U ARE READY TO DRAW YOUR SHAPE.": GOSUB
      600: GOSUB 425
1290  GOSUB 300: IF Z = 33 THEN XI = X:YI =
     Y: HOME : RETURN
1300  GOSUB 350: GOSUB 425: GOTO 1290
1350  GOSUB 225: GOSUB 425: GOSUB 525:I = 0:
      GOSUB 515: VTAB 22: HTAB 1: PRINT A$: IF
      NOT F5 THEN  GOSUB 720: GOSUB 800: IF
     F3 THEN  RETURN
1360 I = 1 - I: REM  TOGGLE
1370  IF F5 THEN  HTAB 1: VTAB 24: CALL  - 8
     68: PRINT "YOU HAVE UP TO "2 * DD - 3 -
     (IP - 16394)" MOVES LEFT.";: IF 2 * DD -
     3 - (IP - 16394) = 0 THEN  RETURN
1380  GOSUB 300: IF Z = 33 THEN  RETURN
1390  IF Z = 8 THEN  GOSUB 450: GOSUB 425: GOTO
     1360: REM  ERASE LAST MOVE
1400  IF Z = 69 AND I = 0 THEN  VTAB 22: HTAB
     1: PRINT "THIS MOVE HAS NO EFFECT ON TH
     E SHAPE.";: FOR J = 1 TO 2000: NEXT J: HTAB
     1: CALL  - 868: PRINT A$: GOTO 1370
1405  GOSUB 350: IF F1 THEN 1370: REM   EVAL
      KEY PRESS FOR NEW X,Y : SET FLAG F1 IF
      ILLEGAL
1410  GOSUB 400: REM  EVALUATE 3 DIGIT BINAR
     Y EQUILVALENT OF KEY PRESS
1420  GOSUB 500: REM  SAVE VECTOR MOVE WITH
     POKE
1430  GOSUB 510: REM  DISPLAY 'ACCUMULATOR'
     WITH TEXT POKES
1440  GOSUB 425: REM   PRINT NEW X,Y COORDS
     TO SCREEN
1450  IF Z > 72 AND Z < 78 THEN  GOSUB 325: REM
       PLOT POINT ON HI-RES FOR APPROPRIATE
     PLOTTING VECTOR
1455  IF  NOT F5 THEN  GOSUB 800: IF F3 THEN
     RETURN
1460  GOTO 1360
1500  GOSUB 230
1510  GOSUB 255
1520  FOR J = 1 TO K: POKE N,( PEEK (SL) + 8
      *  PEEK (SL + 1)):SL = SL + 2:N = N +
     1: NEXT J: POKE N,0: REM  POKE SHAPE 'O
     N TOP OF' VECTORS
1530  GOSUB 260: HGR : HCOLOR= 3: DRAW 1 AT
     XI,YI: GOSUB 265
1540  HOME : VTAB 21: PRINT "DO YOU WISH TO
     SAVE THIS SHAPE (Y/N)?";: GET Z$: IF Z$
      <  > "Y" AND Z$ <  > "N" THEN 1540
1545  IF F5 THEN  RETURN
1550  IF Z$ = "N" THEN  RETURN
1560  FOR J = N + 1 TO N + 2 + .25 * (N - VS
      + 1): POKE J,0: NEXT J: REM   EXPAND S
     HAPE 25% BY ADDING ZEROS AT END
1570 N = J - 1: FOR J = VS + 1 TO N: POKE PA
     , PEEK (J):PA = PA + 1: NEXT J: REM   T
     RANSFER SHAPE FROM TEMPORARY LOCATION T
     O SHAPE TABLE
1580 NS =  PEEK (TS):NS = NS + 1: POKE TS,NS
     : REM  INCREASE # SHAPES IN INDEX BY 1
1590  GOSUB 215: GOSUB 240: REM   POKE DATA
     INTO THIS INDEX LOCATION///INCREMENT IN
     DEX LOCATIION OF NEXT SHAPE
1610  RETURN
1700  GOSUB 600: HOME : VTAB 21: PRINT "USE
     THE GAME PADDLES TO POSITION THE DOTAT
     WHICH POINT THE SHAPE WILL BE DRAWN. PR
     ESS EITHER BUTTON WHEN READY TO VIEW  S
     HAPES.";
1720 X = 140:Y = 80: GOSUB 100:NS =  PEEK (2
     048): HOME : VTAB 21: PRINT "USE THE X
     GAME PADDLE TO VIEW ALL SHAPESIN CURREN
     T TABLE.  PRESS BUTTON WHEN    FINISHED
      VIEWING.": GOSUB 110: GOSUB 120: RETURN
1750  TEXT : HOME : PRINT  TAB( 5);"THE FOLL
     OWING SEQUENCE WILL BE      FOLLOWED IN
      VIEWING A SHAPE."
1755  PRINT : PRINT "1) INPUT SHAPE NO. USIN
     G X GAME PADDLE.": PRINT : PRINT "2) IN
     PUT HCOLOR USING Y GAME PADDLE.": PRINT
     : PRINT "3) MOVE SHAPE TO DESIRED POSIT
     ION.": PRINT : PRINT "4) USE X PADDLE T
     O VARY ROT, AND              Y PADDLE TO V
     ARY SCALE.": GOSUB 975
1760  HOME : PRINT  TAB( 5);"USE THE X GAME
     PADDLE TO CHOOSE YOURSHAPE NO.  PRESS T
     HE PADDLE'S BUTTON    WHEN FINISHED.": GOSUB
     140:SH = S1
1765  PRINT : PRINT  TAB( 5);"INPUT THE HCOL
     OR USING THE Y PADDLE.PRESS ITS BUTTON
     WHEN FINISHED.": GOSUB 160:HC = S1
1770  HGR : HOME : VTAB 21: PRINT  TAB( 5);"
     USE THE GAME PADDLES TO LOCATE THE  POI
     NT WHERE THE SHAPE WILL BE DRAWN.    PR
     ESS EITHER BUTTON WHEN FINISHED.": GOSUB
     100:XI =  INT (X):YI =  INT (Y)
1775  HOME : VTAB 21: PRINT  TAB( 5);"USE PA
     DDLES TO VARY ROTATION (X) ANDSCALE (Y)
     .  PRESS EITHER BUTTON TO STOP.": VTAB
     23: PRINT "SHAPE #"SH SPC( 3)"HCOLOR="H
     C SPC( 3)"X="XI SPC( 3)"Y="YI: GOSUB 17
     4: RETURN
1800  FOR J = N - VS + 1 TO DD:N = N + 1: POKE
     N,0: NEXT J
1810 N = VS + 1:J = 256 *  PEEK (MS) +  PEEK
     (LS) + TS: FOR K = 1 TO DD - 1: POKE J,
      PEEK (N):J = J + 1:N = N + 1: NEXT K: RETURN
3500  GOSUB 1000
3501  GOSUB 1060: REM  ENTRY FOR ADDING TO E
     XISTING TABLE
3502  HOME : GOSUB 770: GOSUB 975: IF F2 THEN
      RETURN
3505  GOSUB 270: HOME : TEXT : PRINT RS" SHA
     PES MAY BE ADDED TO THE CURRENT": PRINT
     "TABLE WHICH CONTAINS "; PEEK (2048);"
     SHAPES.": GOSUB 975
3510  IF  NOT RS THEN 3575
3515  HOME : PRINT  TAB( 5);"DO YOU WISH TO
     DRAW A SHAPE": PRINT "Y/N?";: GET Z$: IF
     Z$ <  > "Y" AND Z$ <  > "N" THEN 3515
3520  IF Z$ = "N" THEN 3575
3525  GOSUB 1200
3530  GOSUB 1350
3535  GOSUB 1500
3540  GOTO 3502
3575  RETURN
3650  GOSUB 8000: ONERR  GOTO 20000
3660  PRINT : PRINT D$"BLOAD"NA$",A"TS: GOSUB
     270: GOSUB 212: GOSUB 203: GOSUB 700: POKE
     216,0: RETURN
3670  HOME : PRINT  TAB( 5);"YOUR FILE NAME
     LENGTH IS ZERO.  DO  YOU STILL WISH TO
     BLOAD A SHAPE TABLE   FROM DISKETTE (Y/
     N)?": GET Z$: IF Z$ <  > "Y" AND Z$ <  >
     "N" THEN 3670
3680  IF Z$ = "N" THEN  RETURN
3690  IF Z$ = "Y" THEN 3650
4000  HOME : IF  PEEK (TS) > 0 THEN 4100
4010  PRINT  TAB( 5);"PRESS THE NUMBER OF YO
     UR CHOICE.": PRINT : PRINT "1) DRAW SHA
     PES/CONSTRUCT A SHAPE TABLE.": PRINT "2
     ) BLOAD A SHAPE TABLE THAT HAS BEEN
        CONSTRUCTED WITH THIS ROUTINE.": PRINT
     "3) QUIT."
4014  GET Z$: IF  VAL (Z$) < 1 OR  VAL (Z$) >
     3 THEN  HOME : GOTO 4010
4016  IF Z$ = "3" THEN 30000
4020  ON  VAL (Z$) GOSUB 3500,3650
4030  GOTO 4000
4100  HOME : PRINT  TAB( 5);"PRESS THE NUMBE
     R OF YOUR CHOICE.": PRINT
4105  PRINT : PRINT "1) DISPLAY SHAPES IN CU
     RRENT TABLE.": PRINT : PRINT "2) ADD SH
     APES TO CURRENT TABLE.": PRINT : PRINT
     "3) CHANGE A SHAPE IN CURRENT TABLE."
4110  PRINT : PRINT "4) BSAVE CURRENT TABLE
     TO DISKETTE.": PRINT : PRINT "5) DELETE
      TABLE CURRENTLY IN MEMORY.": PRINT : PRINT
     "6) QUIT.": PRINT
4120  GET Z$: IF  VAL (Z$) < 1 OR  VAL (Z$) >
     6 THEN 4100
4130  IF Z$ = "6" THEN 30000
4150  HOME : ON  VAL (Z$) GOSUB 5200,5400,56
     00,5800,6000
4160  GOTO 4000
5200  TEXT : HOME : IF  PEEK (TS) = 0 THEN  PRINT
     "THERE ARE NO SHAPES IN TABLE.": GOSUB
     975: RETURN
5205  GOSUB 265: PRINT  TAB( 5);"PRESS THE N
     UMBER OF YOUR CHOICE.": PRINT : PRINT "
     1) VIEW ALL SHAPES.": PRINT : PRINT "2)
      VIEW ONLY ONE SHAPE.": PRINT : PRINT "
     3) RETURN TO MAIN MENU."
5210  GET Z$: IF  VAL (Z$) < 1 OR  VAL (Z$) >
     3 THEN 5200
5215  IF Z$ = "3" THEN  RETURN
5220  ON  VAL (Z$) GOSUB 1700,1750: GOTO 520
     0
5400  GOSUB 3501: RETURN
5600  TEXT : HOME : IF  PEEK (TS) = 0 THEN  PRINT
      TAB( 5);"THERE IS NO TABLE CURRENTLY I
     N      MEMORY.": GOSUB 975: RETURN
5610  PRINT  TAB( 5);"THERE ARE " PEEK (2048
     )" SHAPES IN TABLE.": INPUT "    ENTER
     THE NUMBER OF THE SHAPE YOU   WISH TO C
     HANGE, OR A ! TO RETURN TO THE MAIN MEN
     U.";SH$: IF SH$ = "!" THEN F5 = 0: RETURN

5620 SH =  VAL (SH$): IF SH < 1 OR SH >  PEEK
     (TS) THEN 5600
5630 F5 = 1: GOSUB 210: GOSUB 1200: GOSUB 13
     50: GOSUB 1500
5640  IF Z$ = "Y" THEN  GOSUB 1800
5660  GOTO 5600
5800  IF  PEEK (TS) = 0 THEN  PRINT "THERE A
     RE NO SHAPES IN TABLE.": GOSUB 975: RETURN

5805  PRINT "IF YOU WISH TO SAVE THIS TABLE
     ON A      DIFFERENT DISKETTE, PUT IT IN
     THE DRIVE AT THIS TIME.": PRINT : PRINT
     "PUT THE UTILITY DISKETTE BACK INTO THE
       DRIVE AFTER THE DISK DRIVE'S RED LIGH
     T  GOES OFF.": GOSUB 975: GOSUB 8000: PRINT

5810  PRINT : PRINT D$"BSAVE"NA$",A"TS",L"PA
      - TS: RETURN
5820  HOME : PRINT  TAB( 5);"YOUR FILE NAME
     LENGTH IS ZERO.  DO  YOU STILL WISH TO
     SAVE THE SHAPE TABLE  THAT IS CURRENTLY
      IN MEMORY (Y/N)?";: GET Z$: IF Z$ <  >
     "Y" AND Z$ <  > "N" THEN 5820
5830  IF Z$ = "N" THEN  RETURN
5840  GOTO 5800
6000  HOME : PRINT  TAB( 5);"TYPE THE WORD "
     ;: FLASH : PRINT "DELETE";: NORMAL : PRINT
     " TO DESTROY": PRINT "THE SHAPE TABLE T
     HAT IS CURRENTLY IN    MEMORY.  TYPE ";
     : FLASH : PRINT "SAVE";
6002  NORMAL : PRINT " IF YOU DO NOT WISH TO
     DESTROY THE SHAPE TABLE THAT CURRENTLY
      IS IN MEMORY."
6005  PRINT : INPUT "     PRESS THE RETURN K
     EY AFTER YOUR   CHOSEN ENTRY -> ";Z$
6010  IF Z$ <  > "DELETE" AND Z$ <  > "SAVE"
      THEN 6000
6020  IF Z$ = "DELETE" THEN  RUN
6030  RETURN
8000  HOME : PRINT  TAB( 5);"ENTER THE NAME
     OF THE TABLE, THEN   PRESS RETURN.  THE
      TOTAL LENGTH CAN NOT EXCEED 30 CHARACT
     ERS, AND THE FIRST     CHARACTER MUST B
     E A LETTER."
8005 NA$ = "":X = 2:Y = 6: HTAB X: VTAB Y
8010  GET Z$
8020  IF  LEN (NA$) = 0 AND  ASC (Z$) < 65 OR
      LEN (NA$) = 0 AND  ASC (Z$) > 90 THEN
      VTAB 10: HTAB 1: PRINT "THE FIRST CHAR
     ACTER MUST BE A LETTER.";: FOR I = 1 TO
     1500: NEXT I: HTAB 1: CALL  - 868: HTAB
     X: VTAB Y: GOTO 8010
8030  IF Z$ = "," THEN  VTAB 10: HTAB 1: PRINT
     "DO NOT USE ANY COMMAS";: FOR I = 1 TO
     1500: NEXT I: HTAB 1: CALL  - 868: HTAB
     X: VTAB Y: GOTO 8010
8040  IF  ASC (Z$) = 8 AND  LEN (NA$) > 1 THEN
     X = X - 1: HTAB X: CALL  - 868:NA$ =  LEFT$
     (NA$, LEN (NA$) - 1): GOTO 8010
8050  IF  ASC (Z$) = 8 AND  LEN (NA$) = 1 THEN
     X = X - 1: HTAB X: CALL  - 868:NA$ = ""
     : GOTO 8010
8055  IF  ASC (Z$) = 13 OR  LEN (NA$) > 29 THEN
      RETURN
8060  PRINT Z$;:NA$ = NA$ + Z$:X = X + 1: HTAB
     X: GOTO 8010
8070  IF  ASC (Z$) = 13 THEN  RETURN
20000 ER =  PEEK (222):LN =  PEEK (218) +  PEEK
     (219) * 256
20010  IF LN = 3660 THEN 21000: REM   FILE N
     OT FOUND ERROR WHEN ATTEMPTING TO LOAD
     A SHAPE TABLE
20020  IF ER = 11 AND LN = 5810 THEN  PRINT
     "FIRST CHARACTER IN FILE NAME MUST BE
       A LETTER, AND NO COMMAS MAY APPEAR IN
        THE NAME.  PRESS ANY KEY TO CONTINUE
     .": GET Z$:Z$ = "4": GOTO 4150
20050  STOP
21000  POKE 34,7: HOME : PRINT  TAB( 5);"YOU
     R INPUT FILE NAME DOES NOT EXIST ON DIS
     KETTE.  DO YOU WISH TO SEE A      CATAL
     OG LISTING OF THE DISKETTE THAT IS CURR
     ENTLY IN THE DRIVE (Y/N)?": POKE 34,0
21010  GET Z$: IF Z$ <  > "Y" AND Z$ <  > "N
     " THEN 21000
21020  IF Z$ = "N" THEN  GOTO 21050
21030  PRINT : PRINT D$"CATALOG"
21040  PRINT : PRINT  TAB( 5);"PRESS ANY LET
     TER TO CONTINUE.": GET Z$
21050  POKE 216,0: GOTO 4000
30000  END
```

©

NEW GENERAL LEDGER SYSTEM

for ATARI* 800

$149.95 Microsoft Base

CHART OF ACCOUNTS
TRIAL BALANCE
INCOME STATEMENT
BALANCE SHEET

$219.95 - includes Microsoft Compiler

VISA - MASTER CHARGE - CHECK - MONEY ORDER

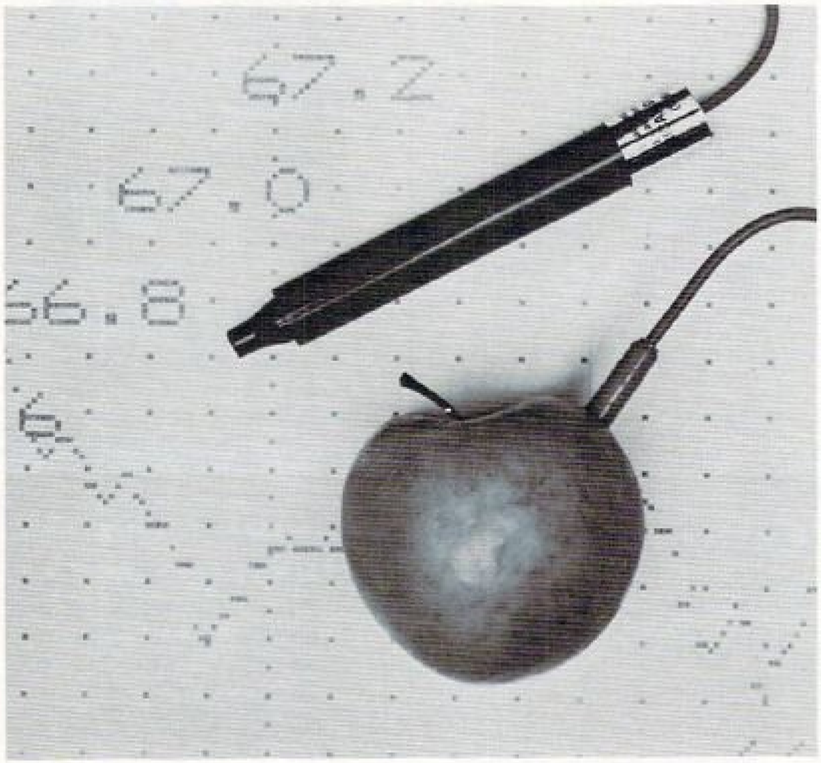
FCC inc.

4712 CHASTANT ST.
METAIRIE, LA. 70002
(504) 454-2421

*Trademark ATARI, INC.

Micro Math, priced at $50, comprises 6 program suites, each containing 4 programs which are available on two cassettes. Subjects covered include algebra, geometry, differentiation-calculus, and statistics.

PM International
P.O. Box 87
Buckfield, ME 04220
(207) 336-2500


The Nth Degree digital temperature probe for Apple II.

Measure Temperatures With Apple

A temperature measuring and control system is now available as a peripheral to the Apple II computer.

The device accurately measures temperatures between -60 and +105 degrees Celsius. Changes of temperature as small as 1/100th of a degree may be detected. The system, produced by American Data Cable, uses a hand-held probe the size of a marking pen and attaches to the Apple II computer by a 6-foot cable. The system software is on a DOS 3.3-compatible, 5.25-inch diskette.

The system, called the Nth Degree, displays temperature readings in either Celsius, Farenheit or Kelvin. An optional program displays or can print out a continuous record of temperature changes in a "strip chart" format.

Accuracies of .01 degree may be obtained, and the probe can be re-calibrated for maximum accuracy over any temperature range. In addition, the system may be programmed for alarms and set points.

The probe can measure reaction rates and temperatures of reactants, and can monitor heating and cooling apparatus. The disk accepts up to 16 years' worth of temperature readings (taken one time an hour) or can store continuous samplings of temperatures taken once a second for 36 hours.

As many as 256 probes may be attached to one system. Temperature readings are made simply by placing the lightweight probe against the surface to be measured. When not in contact with a specific material, the probe measures the temperature of the air.

The model 551A probe with software, Apple II interface adapter and user's manual sells for $129.

American Data Cable, Inc.
2864 Ray Lawyer Dr., No. 205-352
Placerville, CA 95667
(916) 622-3465

English As A Second Language

The Soft Spot is now marketing *Teachers' Friend*, a program that teaches English as a second language (ESL) to students who can read English at the second-grade level. This 80-lesson curriculum, developed and refined over a two-year period, sells for $15 per lesson. The lessons can be used independently of one another. Students can go right to the lesson they need, when they need it, without going through all lessons in sequence.

PERSONAL PERIPHERAL PRODUCTS *presents:*
SPEAKEASY

VIC-20 SPEECH **VIC-20 SPEECH**

CARTRIDGE & "VOCAL CHORD" SOFTWARE $79.95
Watch major software houses for products which are decoded for SPEAKEASY

ALSO: BARE BONES BOARDS

16K Ram Expander For VIC-20 . . . . Kit $49.95
Assembled & Tested $64.95

3 Slot Expansion Board for VIC-20
Switched and Fused . . . . Kit $24.95
Assembled & Tested $34.95

ADD $2.00 Total Order Handling/Ill Residents Add 6% Sales Tax

PERSONAL PERIPHERAL PRODUCTS
P.O. BOX 3423
FOX VALLEY MALL
AURORA, IL 60505 • (312) 961-2347
VIC IS A TRADEMARK OF COMMODORE

"You'll find my KoalaPad™ Touch Tablet, and KoalaWare™ software a natural profit builder."
Available for Apple,® Atari,® Commodore,® and IBM® personal computers.

Introducing the KoalaPad.™ At less than $125 – with software – it's the friendliest, least expensive graphics pad on the market.

Your Atari,® Commodore,® Apple,® and IBM® customers will love it. They'll find it faster than paddle controllers. More flexible than joysticks. And much easier to use than a keyboard.

Plus it's compatible with most of their game and educational software. And most KoalaPad sets come with a graphics program called Micro-Illustrator.™ The most exciting way yet to create colorful, high resolution computer graphics. At the touch of a finger.

There are KoalaWare programs for computer fans of all ages. Dancing Bear,™ the funny, furry computer cabaret. Spider Eater,™ the lively music learning adventure. And Spellicopter,™ the action-packed spelling game.

Selling Koala products is as easy as falling out of a tree. Particularly with our 5% co-op allowance, color p.o.p. materials, and traffic-building promotions. Not to mention our extensive national advertising featuring, ahem, yours truly.

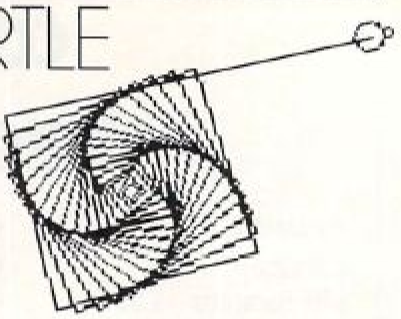
For the name of your nearest Koala distributor, call toll free 800-227-6703, (in California, 800-632-7979). Or write to Koala Technologies Corporation, 4962 El Camino Real, Los Altos, CA 94022.

Koala
Technologies Corporation

We make computing more personal.™

Trademarks: KoalaPad, KoalaWare, Spider Eater and Dancing Bear are trademarks of Koala Technologies Corp. Micro-Illustrator is a trademark of Island Graphics, and Spellicopter is a trademark of DesignWare Inc.

FRIENDS OF THE TURTLE



David D. Thornburg, Associate Editor

PILOT And Logo – A Tale Of Two Languages

PILOT and Logo are two of the most popular user-friendly computer languages available for personal computers. Because Atari PILOT and Apple SuperPILOT both contain a powerful turtle graphics environment, many people wonder if PILOT might not be a substitute for Logo.

As I will show, Logo and PILOT are quite different languages. Although they can be used for many of the same applications, each language has special features that make it more appropriate for some applications than for others. The goal of this article is to provide enough information about both languages to aid someone who is trying to decide which to use. I will assume that you are already familiar with turtle graphics.

PILOT

PILOT stands for Programmed Inquiry, Learning Or Teaching. It was so named by its developer, John Starkweather, because he wanted to create a programming language that easily allowed teachers to generate computer-aided instructional materials. Research in the late 1960s by Dean Brown showed that PILOT was also a good programming language for children.

The key to PILOT's appeal is its simple command structure and powerful ability to manipulate text-oriented material. At its core, PILOT has only eight commands, yet these eight commands allow the creation of quite sophisticated programs. The core commands for PILOT are shown below:

PILOT Command	Function
T:	*T*ypes text and variables on the screen.
A:	*A*ccepts input from the keyboard.
M:	*M*atches words or phrases against the result of the most recent accept command.
J:	*J*umps execution to a label.
U:	*U*ses a labeled procedure.
C:	*C*omputes the value of a variable.
R:	allows *R*emarks to be added to a procedure.
E:	*E*nds a program or procedure.

Notice that none of these commands has anything to do with graphics. The incorporation of turtle graphics in PILOT is a fairly recent event. Also, most versions of PILOT have additional text manipulation commands that add significantly to its power.

Core PILOT's most powerful command is M:, the match command. To see why this command is so powerful, consider the following PILOT procedure:

```
*QUESTION1
  T: WHAT GROWS ON TREES?
  A:
  M: MOSS, LEAVES, BUGS, INSECTS, NEEDLES
     TY: YOU ARE CORRECT
     TN: ARE YOU SURE? LET'S TRY AGAIN.
     JN: *QUESTION1
  E:
```

This PILOT procedure works in the following way. First, a question is typed on the screen. The user then types a response that is saved in the "accept buffer." The match command then checks to see if any of the words, MOSS, LEAVES, etc., appear anywhere in this buffer. If there is a match, a "yes flag" (Y) is set to be true and a "no flag" (N) is set to be false. The execution of any PILOT command can be made conditional on the status of these flags by entering Y or N after the command name. For example, the command TY: will print on the screen only if the yes flag is true. The JN: command causes the procedure to be used over again if the user's response is *not* matched.

As a result of PILOT's ability to manipulate words and phrases, many of the early uses of PILOT by children involved the creation of word games and "poetry generators."

What About PILOT Graphics?

As mentioned, graphics is a recent addition to PILOT. Turtle graphics is incorporated through the use of special commands. In Atari PILOT, for example, this command is GR: followed by specific graphics instructions. The fundamental graphics commands allow the turtle to be moved in its present heading or to have its heading changed.

Here's a list of the more commonly used Atari PILOT graphics commands:

PILOT Command	Function
GR: DRAW x	Draws a line of length x in the present heading.
GR: TURN x	Rotates the turtle by x degrees.
GR: PEN UP	Raises the turtle's pen.
GR: PEN YELLOW	Sets the pen color to yellow and sets the pen down.
GR: GOTO x, y	Moves the turtle to absolute coordinates x,y.
GR: TURNTO x	Rotates the turtle to absolute orientation of x degrees measured to the right of straight up.

These commands (and several others) allow the creation of procedures that draw complete figures. For example, the PILOT procedure shown below draws a square 50 units on a side:

```
*SQUARE
  GR: 4(DRAW 50 ; TURN 90)
E:
```

To use this procedure, one would type:

```
U: *SQUARE
```

Logo

Logo is a computer language that was designed by Seymour Papert to be an easy, yet powerful tool which would let children use the computer to explore topics on their own. While designed to be used by children, Logo is a user-friendly version of the tremendously powerful language, LISP. Since LISP is the language of choice for many researchers in the field of artificial intelligence, clearly Logo is a programming language for adults as well.

The key to Logo's appeal is its simple syntax (compared with LISP) and its ability to manipulate data structures called *lists*. A list is a collection of words, Logo commands, numbers, or other lists. Logo allows lists to be constructed, modified, examined, reordered, and (if the list consists of Logo procedures or primitive commands) executed. Here are some core Logo commands which are comparable to the core PILOT commands:

Logo Command	Function
PRINT	Prints a list of text on the screen.
READLIST	Reads a list from the keyboard.
MEMBERP	A predicate that matches a word against the elements of a list.
MAKE	Assigns (or "binds") a number, word, or list to a variable named by a word.
END	Ends a procedure.
FIRST	Returns the first element of a list.
BUTFIRST	Returns all but the first element of a list.
LAST	Returns the last element of a list.
BUTLAST	Returns all but the last element of a list.

Notice that none of these commands has anything to do with graphics. Turtle graphics was incorporated into Logo after the language had been in use for a while. The list of Logo primitives shown above is quite incomplete, but it allows us to build a procedure comparable to the QUESTION1 procedure we wrote in PILOT:

```
TO QUESTION1
    PRINT [WHAT GROWS ON TREES?]
    MAKE "ANSWER READLIST
    TEST MEMBERP FIRST :ANSWER [MOSS
        LEAVES BUGS INSECTS NEEDLES]
        IFTRUE [PRINT [YOU ARE CORRECT]]
        IFFALSE [PRINT [ARE YOU SURE? LET's
        TRY AGAIN]
            QUESTION1]
END
```

This procedure performs a function similar to that of the PILOT procedure except that it only looks to see if the first word on the answer is contained in the answer list. The commands following the words IFTRUE are executed only if the result of TEST is true. If the result is false, the commands following IFFALSE are executed instead. Notice that a Logo procedure is treated just as if it were a Logo primitive. To execute the procedure QUESTION1, you merely type its name.

As with PILOT, many of the early uses of Logo by children involved the creation of word games and poetry.

What About Logo Graphics?

A list of the more common Logo turtle graphics commands is shown below:

Logo Command	Function
FORWARD x	Draws a line of length x in the present heading.
RIGHT x	Rotates the turtle by x degrees.
PENUP	Raises the turtle's pen.
PENDOWN	Sets the pen down.
SETPOS x y	Moves the turtle to absolute coordinates x,y.
SETHEADING x	Rotates the turtle to absolute orientation of x degrees measured to the right of straight up.

These commands (and several others such as BACK and LEFT) allow the creation of procedures that draw complete figures. For example, the following procedure draws a square of any size:

```
TO SQUARE :SIZE
    REPEAT 4 [FORWARD :SIZE RIGHT 90]
END
```

To use this procedure to draw a square 50 units on a side, one would enter:

```
SQUARE 50
```

Differences Between Logo And PILOT

The previous sections have suggested that PILOT and Logo are similar in application areas and syntax. In fact, there are some major differences between the languages that may cause one to be clearly the language of choice for a particular task.

For example, PILOT makes it very easy to create programs in which the contents of variables are printed along with text. Also, the match command will compare each element of a list with the entire response. In Logo, you would have to write a procedure to do this.

Another important feature of PILOT is its compactness. Most Logo implementations require large amounts of RAM. Most (but not all) versions of PILOT will operate in 16K of RAM with plenty of space left for the user's program.

In terms of overall symbol manipulation, Logo is the more powerful of the two languages. The ability to write programs that generate other programs is of great utility when constructing environments that "learn from experience." The fact that user-defined procedures are treated exactly as if they were Logo primitives gives Logo a feature called *extensibility*. This means that you can add new words to Logo's vocabulary (as we did with QUESTION1 and SQUARE). There is no need in Logo for the *jump* or *use* commands. To execute a procedure, you just type its name.

Logo also supports *local variables*. This means that the value associated with a variable is assigned to the specific procedure (and level) in which it is used. This allows you to write procedures that use themselves recursively. For more information on this topic, you might want to read the "Friends of the Turtle" columns on recursion that appeared a few months back.

Logo's turtle graphics commands are, perhaps, easier to grasp than PILOT's, but there are indications that this will not always be the case as new versions of PILOT are likely to become more "Logo-like."

Apart from these differences, Logo and PILOT both encourage a procedure-oriented programming style that makes complex programs easy to read and correct.

I use both languages regularly and find that I would be reluctant to abandon either one. Your application areas might indicate that one of these languages has a clear advantage over the other. No matter which you choose, you will be using a language that allows the creation of very sophisticated and powerful programs.

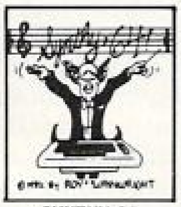
Notes From All Over

I have just heard from my Argentinian friend, Horacio Reginni, who has just started the Asociacion Amigos de Logo (Logo Friends Association) to promote the development of Logo centers, sponsor meetings, and spread information about Logo all over the world. The association can be reached at 2969 Salguero St., Buenos Aires, 1425 Argentina. True Logophiles will be interested in attending their first International Logo Conference in Buenos Aires on September 16-18. Registration is only $25. As for the air fare ....

©

GET THE MOST OUT OF YOUR COMMODORE-64 WITH SOFTWARE FROM ABACUS

SCREEN GRAPHICS 64
- Adds 24 commands to BASIC.
- Plot points, lines, boxes, circles and fill in hires and multicolor.
- Define and animate sprites easily.
- Includes demos, tutorial and manual.
- $24.95 FOR TAPE
- $27.95 FOR DISK



SYNTHY 64
- Full featured music and sound synthesizer.
- Easy entry and editing of notes and commands.
- Control ASDR, filters, waveforms etc.
- Includes sample music and manual.
- $29.95 FOR TAPE
- $32.95 FOR DISK

CHARTPAK 64
- Create Pie, Bar and Line charts in high resolution.
- Enter, edit, save and recall to/from disc.
- Choose any of 8 chart formats and design charts interactively.
- Produce hard copy onto 1515, 1525 or Epson printer.
- $42.95 DISK ONLY

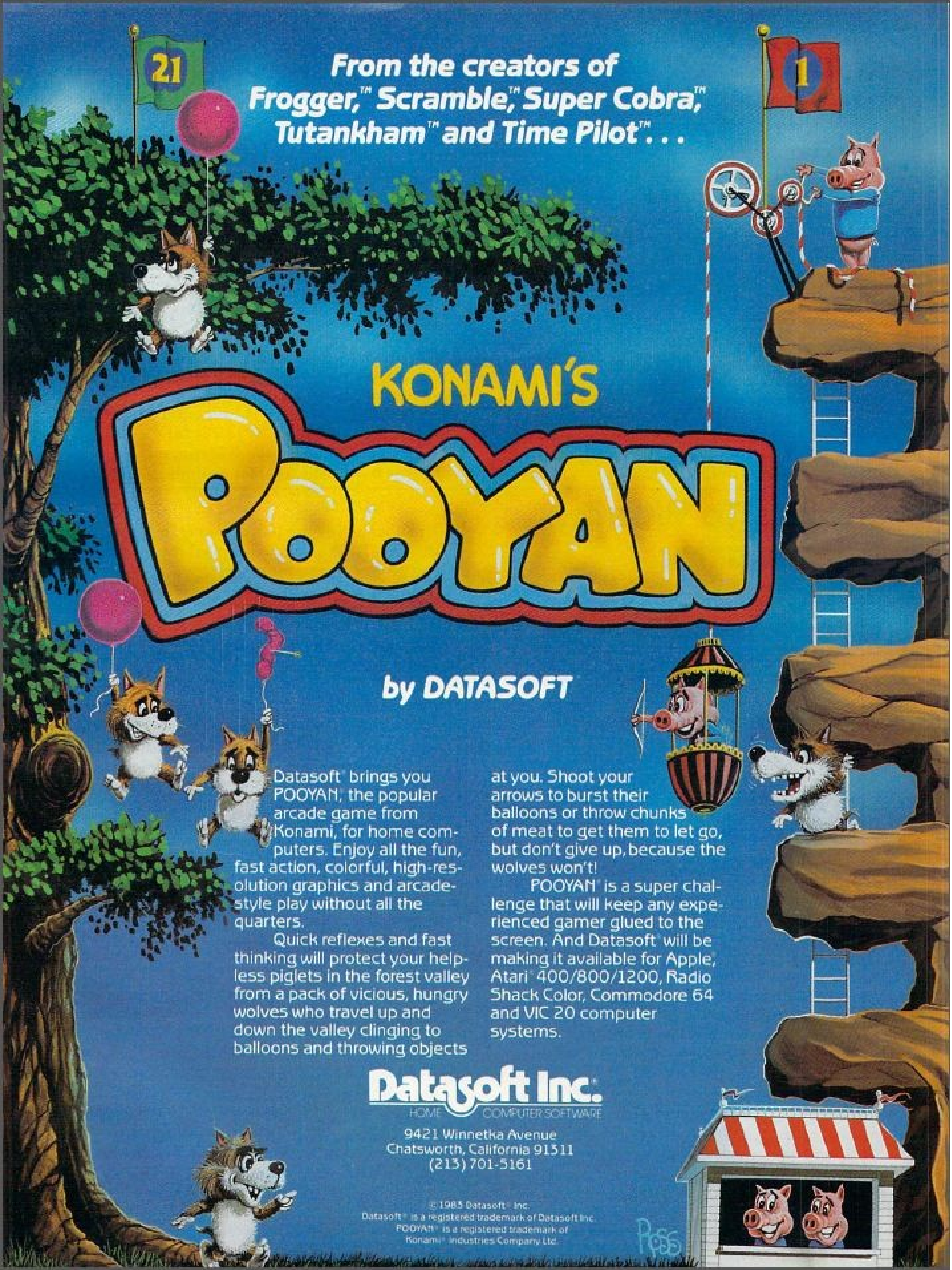
ULTRABASIC 64
- Turtle, hires, multicolor and sprite graphics.
- Sound and sound effects.
- Screen copy to 1515, 1525 or Epson printers.
- Includes demos, tutorial, manual.
- $39.95 FOR TAPE
- $42.95 FOR DISK

TO ORDER NOW
PLEASE WRITE: ABACUS SOFTWARE
P.O.BOX 7211, Grand Rapids, MI 49510

For postage & handling, add $1.50 (U.S. and Canada), add $3.00 for foreign. Make payment in U.S. dollars by check, money order or charge card. **FOR IMMEDIATE SERVICE PHONE 616/241-5510**

FREE CATALOG
Ask for a listing of other software for your Commodore-64 or VIC-20.

YOU CAN COUNT ON
Abacus Software



From the creators of Frogger,™ Scramble,™ Super Cobra,™ Tutankham™ and Time Pilot™. . .

KONAMI'S

POOYAN

by DATASOFT®

Datasoft® brings you POOYAN,® the popular arcade game from Konami, for home computers. Enjoy all the fun, fast action, colorful, high-resolution graphics and arcade-style play without all the quarters.

Quick reflexes and fast thinking will protect your helpless piglets in the forest valley from a pack of vicious, hungry wolves who travel up and down the valley clinging to balloons and throwing objects at you. Shoot your arrows to burst their balloons or throw chunks of meat to get them to let go, but don't give up, because the wolves won't!

POOYAN® is a super challenge that will keep any experienced gamer glued to the screen. And Datasoft® will be making it available for Apple,® Atari® 400/800/1200, Radio Shack Color, Commodore 64 and VIC 20 computer systems.

Datasoft Inc.®
HOME COMPUTER SOFTWARE

9421 Winnetka Avenue
Chatsworth, California 91311
(213) 701-5161

PLOTting On The Apple

Thomas P. Anderson

How to plot and handle the screen on Apple's high resolution screen. Requires 16K RAM. This machine language routine simplifies screen graphics.

This little study of mine, which began about three months ago, first started after I had written a short BASIC program to plot pictures on the high resolution screen with four lines of text at the bottom. The entire screen memory had to be saved to store the picture on the disk. To avoid this waste of space, I decided to find out the memory locations of the four bottom screen lines. I could then devise a method of saving all screen memory except for those four lines.

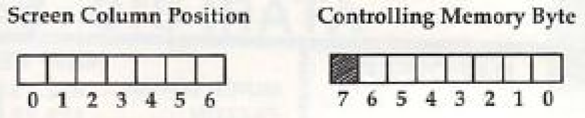
I quickly found the necessary addresses, but in the process I also noticed how strangely the screen memory was laid out. There had to be a way of decoding the inconsequent order of screen memory, so that a specific point on the screen could be referenced easily.

How does Applesoft do it? I found absolutely no documentation of this subject. I could have waded through about 8K of disassembled code and still not found the answer, so I was on my own in figuring this one out. In this article, I am relating to you what I have found out about PLOTting on the high resolution screen in machine language.

Base Addresses

First of all, a review of the hi-res screen layout. The screen has a resolution of 192 lines by 280 dots. The lines are referenced by the decimal values 0-191, and the dots are referenced by the decimal values 0-279. The position of (0,0) is in the upper left-hand corner.

Seven consecutive dots of a line are controlled by the value stored in one byte of memory, so 40 bytes are required to control one line. These 40 bytes, referenced by the decimal values 0-39, I will call the column position. The zero column position is the base address in memory of the line. To see what values are necessary to turn on a dot, we have to look at the bit patterns of the controlling byte.



Shown above are seven consecutive dots on the screen and the controlling byte. If bit three of the controlling byte is on, then the dot in position three within the column will be on. Bit seven of the controlling byte will be zero for this article, since I am not concerned here with manipulating the screen colors.

What To Calculate

Once I had reviewed the basics of screen memory, my problem became defined for me. The routines I had to write would take two decimal values: 1) a line number in the range of 0-191, and 2) a dot position in the range of 0-279. From these values the routines would calculate:

1) The 16-bit hexadecimal *base address*,
2) The *column position* in the range of 0-39, and
3) The *dot position within* the *column* in the range of 0-6.

I will explain the calculation of the base address first, since it is the complicated one. To understand this, I had to know what all the possible base addresses of page two could be. I used page two during the testing because PLOTting on page one wrote over my source file in memory, and it had to be reloaded after every test.

All base addresses lie within the range of $4000-$7FD0. This means that the high byte of our address will be in the range of $40-$7F, and the only possibilities for the low byte are $00, $28, $50, $A8, or $D0. To see how this works exactly, I assigned variables to the bit positions of the line number.

Since we know the maximum and minimum value of the base address, we know that certain bits of the base address will always be off or on no matter what the line number is. Shown below are the assigned variables of the line and the starting framework of the base address.

Line Number

A	B	C	D	E	F	G	H
7	6	5	4	3	2	1	0

Base Address

0	1	0	?	?	?	?	?	?	?	?	?	?	0	0	0
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
High Byte								**Low Byte**							

Figuring Bit Positions

The steps that follow are the ones that I used to figure the bit position of the line that would determine the value of a bit position in the base address.

1. Choose a questionable bit position of the base address.
2. Determine all the possible values of that byte if the bit is on.
3. Determine all the possible values of that byte if the bit is off.
4. Determine all the possible values of the line number based on the values found in Step 2.
5. Determine all the possible values of the line number based on the values found in Step 3.
6. By examining the binary values of the line numbers, the bit patterns are easily seen. There will be one bit position in the values from Step 4 that is always the complement of that same bit position in the values from Step 5. Therefore, this bit position of the line number is the determining bit for the questionable bit in Step 1.

Using these steps for all questionable bits of the base address, I ended up with the representation of the base address as shown below.

Base Address

0	1	0	F	G	H	C	D	E	A	B	A	B	0	0	0
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
High Byte								**Low Byte**							

Now that I had figured out the starting and ending representations, I wrote the routine HBASCALC to perform this operation. This routine is shown in Program 1. The routine is entered with the line number in location HCV and exits the routine with the base address in locations HBASL and HBASH. The documentation explains the process and shows how the variable representations are manipulated by each instruction. This routine can be easily changed to use page one by changing the instruction ORA#$40 to ORA#$20 and setting the appropriate soft switches for page one.

That may have been a bit complex; but now that you have the HBASCALC routine, you won't have to follow those steps as I did.

Calculating the column position and the dot position within the column was a simpler task for me. Since there are 280 dots to represent with 40 bytes of memory, I needed to divide the dot position by seven. The quotient would be the column position, and the remainder would be the dot position within the column. At first I used the standard 16-bit by 16-bit division routine, but this process seemed a little slow for PLOTting.

So I tried another routine which simply subtracted seven from the dot position until it went to less than zero. An index register was used to count the number of times seven could be subtracted, which gave me the column position, and then adding seven back to the now negative dot position gave me the dot position within the column.

Being unsure of the speed of this routine, I calculated the time required by each routine to plot 280 positions, one full line, and I found that the second routine used about 120,000 machine cycles less than the standard routine. The faster routine is called DIVIDE and is shown with the demo program (Program 2).

Plotting A Grid

This division process is the most time consuming aspect of hi-res plotting. I learned quickly that if the main driving routine consists of nested loops, as Hi-res Grid Demo does, then the division routine should be used in the outer routine, or the entire routine will be greatly slowed. If the division is performed in the inner loop, it will be executed for every dot plotted. If the division is performed in the outer loop, it will be executed only when the dot position changes.

Program 2, Hi-res Grid Demo, will accept input from the user in the range of 1-9. It then draws the grid with the number input as the number of spaces between each line of the grid. The program PLOTs the grid until it goes off the edge of the screen, and then it goes back and erases the excess plots to make a neater appearing grid. The program will terminate with a CONTROL-C.

Hi-res Grid Demo is fairly simple, but its purpose is to show you the basic routines used in PLOTting with machine language. Perhaps it will stimulate you to look further to other possibilities, such as color PLOTting, line drawing routines, animation, and faster game design than BASIC will allow. I know it has me working on other ideas. As for my original objective, the saving of hi-res pictures without the four bottom lines, I forgot all about that once I discovered the other interesting possibilities at my fingertips.

Program 1: Base Address Calculation

```
* LISTING 1.
*
*********************************************
*HIRES BASE ADDRESS CALCULATION ROUTINE,PAGE 2*
*********************************************
HBASCALC PHA
         LDA  HCV     ;ABCDEFGH; GET VERT. POS.(0-191).
         ASL          ;BCDEFGH0; SHIFT LEFT UNTIL BITS
         ASL          ;CDEFGH00; "FGH" ARE  IN CORRECT POSITION.
         AND  #$1C    ;000FGH00; TURN OFF ALL BUT BITS "FGH".
         ORA  #$40    ;010FGH00; BIT SIX IS ALWAYS ON.
         STA  HBASH   ;010FGH00; SAVE THIS PORTION.
         LDA  HCV     ;ABCDEFGH; START AGAIN.
         LSR          ;0ABCDEFG; SHIFT RIGHT UNTIL THE
         LSR          ;00ABCDEF; "E" BIT SHIFTS TO CARRY.
         LSR          ;000ABCDE; AND BITS "CD"
         LSR          ;0000ABCD; ARE IN CORRECT POSITION.
         AND  #3      ;000000CD; TURN OFF ALL BUT "CD" AND
         ORA  HBASH   ;010FGHCD; MERGE TO COMPLETE HI-BYTE
         STA  HBASH   ;010FGHCD;
         LDA  HCV     ;ABCDEFGH; WORK ON LO-BYTE.
         AND  #$C0    ;AB000000; TURN OFF ALL BUT BITS "AB".
         PHA          ;AB000000; SAVE IT.
         ROR          ;EAB00000; GET "E" BIT BACK FROM CARRY.
         STA  HBASL   ;EAB00000; SAVE THIS PORTION.
         PLA          ;AB000000; PULL BACK BITS "AB" AND
         LSR          ;0AB00000; SHIFT RIGHT UNTIL THEY
         LSR          ;00AB0000; ARE IN CORRECT POSITION.
         LSR          ;000AB000;
         ORA  HBASL   ;EABAB000; MERGE TO COMPLETE LO-BYTE.
         STA  HBASL   ;EABAB000;
         PLA
         RTS
```

Program 2: Hi-res Grid Demo

```
* LISTING 2
*
********************
* HIRES GRID DEMO *
* BY TOM ANDERSON *
********************
*
* SYSTEM SOFT SWITCHES
*
KBD      EQU  $CØØØ ;READ KEYBOARD
KBDSTR   EQU  $CØ1Ø ;CLEAR KEYBOARD
GRAPHICS EQU  $CØ5Ø ;GRAPHICS MODE
HIRES    EQU  $CØ57 ;HI-RESOLUTION GRAPHICS
PRIMARY  EQU  $CØ54 ;PAGE ONE
ALLGR    EQU  $CØ52 ;FULL SCREEN GRAPHICS
TXTMODE  EQU  $CØ51 ;TEXT MODE
SECOND   EQU  $CØ55 ;PAGE TWO
*
*PAGE ONE LOCATIONS USED
*
CH       EQU  $24   ;TEXT COLUMN POSITION(Ø-39)
CV       EQU  $25   ;TEXT LINE POSITION(Ø-23)
HCV      EQU  $25   ;HIRES LINE POSITION(Ø-191)
HPOSLO   EQU  $26   ;HIRES DOT POSITION(Ø-279)
HPOSHI   EQU  $27
HBASL    EQU  $28   ;HIRES LINE BASE ADDRESS
HBASH    EQU  $29
REMLO    EQU  $2C   ;REMAINDER IN DIVISION ROUTINE
REMHI    EQU  $2D
GRIDSZ   EQU  $2E   ;VALUE OF GRID SIZE
RTMARG   EQU  $2F
```

```
             *
             *MONITOR ROUTINES
             *
             COUT     EQU  $FDED    ;CHARACTER OUTPUT ROUTINE IN MONITOR
             HOME     EQU  $FC58    ;MONITOR ROUTINE TO CLEAR TEXT PAGE
             BASIC    EQU  $3D0     ;VECTOR TO RETURN TO CURRENT BASIC
7000: 20 C2 70  START    JSR  PRDISP   ;DISPLAY INPUT PROMPT
7003: 20 9B 70           JSR  INPUT    ;GET USER INPUT
7006: 20 4A 71           JSR  INHRES   ;INITIALIZE HIRES MODE
             ************************
             * DRAW VERTICAL LINES *
             ************************
7009: A9 00              LDA  #0       ;START AND RESET DOT POSITION TO ZERO
700B: 85 26              STA  HPOSLO
700D: 85 27              STA  HPOSHI
700F: A9 00     VERT     LDA  #0       ;START LINE ZERO
7011: 85 25              STA  HCV
7013: 20 EF 70           JSR  DIVIDE   ;CALCULATE HORIZONTAL OFFSET
7016: 20 21 71  VERT1    JSR  HBASCALC ;CALCULATE LINE BASE ADDRESS
7019: 20 10 71           JSR  DISPLAY  ;TURN ON ONE DOT
701C: E6 25              INC  HCV      ;LINE=LINE+1
701E: A5 25              LDA  HCV
7020: C9 C0              CMP  #192     ;BOTTOM OF SCREEN?
7022: 90 F2              BCC  VERT1    ;NO, GO BACK
7024: 18                 CLC
7025: A5 26              LDA  HPOSLO   ;DOT POSITION=DOT POSITION+GRIDSZ
7027: 65 2E              ADC  GRIDSZ
7029: 90 02              BCC  OVERV
702B: E6 27              INC  HPOSHI
702D: 85 26     OVERV    STA  HPOSLO
702F: A5 27              LDA  HPOSHI   ;END OF LINE?
7031: 4A                 LSR
7032: 90 DB              BCC  VERT     ;NO, GO BACK
7034: A5 26              LDA  HPOSLO
7036: C9 18              CMP  #$18
7038: 90 D5              BCC  VERT
703A: E5 2E              SBC  GRIDSZ   ;RTMARG=LAST DOT POSITION PLOTTED
703C: 85 2F              STA  RTMARG
             **************************
             * DRAW HORIZONTAL LINES *
             **************************
703E: A9 00              LDA  #0       ;START DOT POSITION ZERO
7040: 85 26              STA  HPOSLO
7042: 85 27              STA  HPOSHI
7044: A9 00     HORIZ    LDA  #0       ;START AND RESET LINE TO ZERO
7046: 85 25              STA  HCV
7048: 20 EF 70           JSR  DIVIDE   ;CALCULATE HORIZONTAL OFFSET
704B: 20 21 71  HORIZ1   JSR  HBASCALC ;CALCULATE LINE BASE ADDRESS
704E: 20 10 71           JSR  DISPLAY  ;DISPLAY VALUE TO TURN ON ONE DOT
7051: 18                 CLC
7052: A5 25              LDA  HCV
7054: 65 2E              ADC  GRIDSZ   ;LINE=LINE+GRIDSZ
7056: 85 25              STA  HCV
7058: C9 C0              CMP  #192     ;BOTTOM OF SCREEN?
705A: 90 EF              BCC  HORIZ1   ;NO, GO BACK
705C: E6 26              INC  HPOSLO   ;YES,DOT POS.=DOT POS.+1
705E: D0 02              BNE  OVERH
7060: E6 27              INC  HPOSHI
7062: A5 27     OVERH    LDA  HPOSHI
7064: 4A                 LSR
7065: 90 DD              BCC  HORIZ
7067: A5 26              LDA  HPOSLO
7069: C5 2F              CMP  RTMARG   ;REACHED RIGHT MARGIN?
706B: 90 D7              BCC  HORIZ    ;NO, GO BACK
706D: A5 25              LDA  HCV      ;DETERMINE BOTTOM CUTOFF POINT
706F: E5 2E              SBC  GRIDSZ
7071: 85 25              STA  HCV
7073: A9 00              LDA  #0
```

```
7Ø75: E6 25                      INC  HCV
7Ø77: 2Ø 21 71      CLRBTM       JSR  HBASCALC ;CLEAR UNNESSECARY PLOTS AT
7Ø7A: AØ 28                      LDY  #4Ø      ;BOTTOM OF THE SCREEN
7Ø7C: 88            NXTBYT       DEY
7Ø7D: 91 28                      STA  (HBASL),Y
7Ø7F: DØ FB                      BNE  NXTBYT
7Ø81: E6 25                      INC  HCV
7Ø83: A6 25                      LDX  HCV
7Ø85: EØ CØ                      CPX  #192
7Ø87: DØ EE                      BNE  CLRBTM
7Ø89: AD ØØ CØ      RDKEY        LDA  KBD      ;GRID DRAWN, A CONTROL-C AT THIS
7Ø8C: 1Ø FB                      BPL  RDKEY    ;POINT WILL TERMINATE
7Ø8E: 8D 1Ø CØ                   STA  KBDSTR   ;ANY OTHER KEYSTROKE WILL RESTART
7Ø91: C9 83                      CMP  #$83
7Ø93: DØ Ø3                      BNE  RESTART
7Ø95: 4C A7 7Ø                   JMP  EXIT
7Ø98: 4C ØØ 7Ø      RESTART      JMP  START
                    ***************
                    * USER INPUT *
                    ***************
7Ø9B: AD ØØ CØ      INPUT        LDA  KBD      ;SINGLE KEY INPUT
7Ø9E: 1Ø FB                      BPL  INPUT
7ØAØ: 8D 1Ø CØ                   STA  KBDSTR
7ØA3: C9 83                      CMP  #$83     ;CONTROL-C WILL TERMINATE
7ØA5: DØ ØC                      BNE  DIG      ;NOT CNTRL-C
7ØA7: 2Ø 58 FC      EXIT         JSR  HOME
7ØAA: AD 51 CØ                   LDA  TXTMODE
7ØAD: AD 54 CØ                   LDA  PRIMARY
7ØBØ: 4C DØ Ø3                   JMP  BASIC
7ØB3: C9 B1         DIG          CMP  #$B1     ;IS IT < 1?
7ØB5: 9Ø E4                      BCC  INPUT    ;YES,INVALID GO BACK
7ØB7: C9 BA                      CMP  #$BA     ;IS IT > 9?
7ØB9: BØ EØ                      BCS  INPUT    ;YES,INVALID GO BACK
7ØBB: 29 ØF                      AND  #$ØF     ;MASK OFF 4 MSB'S
7ØBD: 69 Ø1                      ADC  #1
7ØBF: 85 2E                      STA  GRIDSZ ;THIS IS SIZE OF GRID
7ØC1: 6Ø                         RTS
                    *************************
                    * DISPLAY INPUT PROMPT *
                    *************************
7ØC2: AD 51 CØ      PRDISP       LDA  TXTMODE ;SET SWITCHES FOR TEXT MODE PAGE ONE
7ØC5: AD 54 CØ                   LDA  PRIMARY
7ØC8: 2Ø 58 FC                   JSR  HOME     ;CLEAR SCREEN
7ØCB: A9 ØC                      LDA  #12      ;SET DISPLAY FOR HTAB 1Ø,VTAB 12
7ØCD: 85 25                      STA  CV
7ØCF: A9 ØA                      LDA  #1Ø
7ØD1: 85 24                      STA  CH
7ØD3: A2 ØF                      LDX  #15
7ØD5: BD DF 7Ø      NXTCHR       LDA  PROMPT,X ;GET CHARACTER
7ØD8: 2Ø ED FD                   JSR  COUT     ;DISPLAY
7ØDB: CA                         DEX
7ØDC: DØ F7                      BNE  NXTCHR
7ØDE: 6Ø                         RTS
7ØDF: AØ BF AØ      PROMPT       ASC  "   ? )9-1(EZISDIRG"
                    ***********************************
                    *DIVIDE DOT POSITION BY SEVEN *
                    ***********************************
7ØEF: A5 26         DIVIDE       LDA  HPOSLO
7ØF1: 85 2C                      STA  REMLO
7ØF3: A5 27                      LDA  HPOSHI
7ØF5: 85 2D                      STA  REMHI
7ØF7: 38                         SEC
7ØF8: AØ FF                      LDY  #$FF
7ØFA: C8            DIV1         INY
7ØFB: A5 2C                      LDA  REMLO
7ØFD: E9 Ø7                      SBC  #7
7ØFF: 85 2C                      STA  REMLO
71Ø1: A5 2D                      LDA  REMHI
```

```
71Ø3: E9 ØØ              SBC  #Ø
71Ø5: 85 2D              STA  REMHI
71Ø7: 1Ø F1              BPL  DIV1
71Ø9: A5 2C              LDA  REMLO
71ØB: 69 Ø7              ADC  #7
71ØD: 85 2C              STA  REMLO
71ØF: 6Ø                 RTS
            ********************
            * DISPLAY ROUTINE *
            ********************
711Ø: A6 2C     DISPLAY  LDX  REMLO   ;DOT POSITION WITHIN COLUMN(Ø-6)
7112: BD 1A 71           LDA  ONBIT,X ;GET VALUE TO TURN BIT ON
7115: 11 28              ORA  (HBASL),Y ;MERGE WITH VALUE ALREADY THERE
7117: 91 28              STA  (HBASL),Y ;DISPLAY NEW VALUE
7119: 6Ø                 RTS
711A: Ø1 Ø2 Ø4  ONBIT    HEX  Ø1Ø2Ø4Ø81Ø2Ø4Ø
            ****************************************************
            *HIRES BASE ADDRESS CALCULATION ROUTINE,PAGE 2*
            ****************************************************
7121: 48        HBASCALC PHA
7122: A5 25              LDA  HCV
7124: ØA                 ASL
7125: ØA                 ASL
7126: 29 1C              AND  #$1C
7128: Ø9 4Ø              ORA  #$4Ø
712A: 85 29              STA  HBASH
712C: A5 25              LDA  HCV
712E: 4A                 LSR
712F: 4A                 LSR
713Ø: 4A                 LSR
7131: 4A                 LSR
7132: 29 Ø3              AND  #3
7134: Ø5 29              ORA  HBASH
7136: 85 29              STA  HBASH
7138: A5 25              LDA  HCV
713A: 29 CØ              AND  #$CØ
713C: 48                 PHA
713D: 6A                 ROR
713E: 85 28              STA  HBASL
714Ø: 68                 PLA
7142: 4A                 LSR
7143: 4A                 LSR
7144: Ø5 28              ORA  HBASL
7146: 85 28              STA  HBASL
7148: 68                 PLA
7149: 6Ø                 RTS
            **************************
            * CLEAR HIRES PAGE TWO *
            **************************
714A: A9 ØØ     INHRES   LDA  #Ø      ;START LINE ZERO
714C: 85 25              STA  HCV
714E: 2Ø 21 71  SCREEN   JSR  HBASCALC ;NEW BASE ADDR. WHEN HCV CHANGES
7151: AØ 28              LDY  #4Ø     ;NUMBER OF COLUMNS
7153: 88        LINE     DEY          ;COLUMN=COLUMN-1
7154: 91 28              STA  (HBASL),Y ;DISPLAY VALUE ZERO
7156: DØ FB              BNE  LINE    ;COLUMN=ZERO?
7158: E6 25              INC  HCV     ;YES, LINE=LINE+1
715A: A6 25              LDX  HCV
715C: EØ CØ              CPX  #192    ;LAST LINE CLEARED?
715E: DØ EE              BNE  SCREEN ;NO, GO BACK
            ********************************
            * SET SOFT SWITCHES FOR HIRES *
            ********************************
716Ø: AD 57 CØ           LDA  HIRES
7163: AD 5Ø CØ           LDA  GRAPHICS
7166: AD 55 CØ           LDA  SECOND
7169: AD 52 CØ           LDA  ALLGR
716C: 6Ø                 RTS
```

Apple Bytechanger

Wally Hubbard

Did you think that there's no way to put RETURNs into REM statements? Or into PRINT statements? Or to put backspace characters into REM statements?

This machine language search and replace program opens a universe of options like these. Use your imagination after you type in the BASIC listing. This article also throws some light on how BASIC is stored in your computer.

A machine language program can be stored in your Apple computer three ways: (1) by typing on the keyboard; (2) by loading it from cassette or disk; (3) by LOADing a BASIC program and having it POKE the machine language into place. It's the third method that we'll use here.

When you RUN this program, it will ask you to provide information so that it can set itself up for the particular function you have in mind. Once you have done this, you can LOAD another BASIC program without affecting the machine language program. Then to change the new program, type & and press RETURN. The computer will jump to the machine language program, execute it, and return to BASIC.

Search And Replace

This program will search through your BASIC program until it finds a REM statement, then read the information between REM and the end of the line, and change any control-A's to carriage returns. When it reaches the end of the line (or a colon), it goes on to the next line and continues its search for REM statements until it reaches the end of the BASIC program.

You can change it so that it will look for any other command, and change characters that follow on that line, until the end of the line or the colon is encountered.

For example, say you want to make your REM statements easier to read by inserting carriage returns. When you type the REM statement, type a control-A everywhere you want a carriage return. Then, when you're finished, use the & command to execute the machine language program. You'll see the results when you list your program.

As another example, suppose your printer requires the Escape character to access special functions. It is possible to type your BASIC program with control-E's in place of the Escape character, then later run the machine language program to make a switch.

BASIC Tricks

BASIC uses some space-saving tricks to store a program. For one, it converts commands into tokens. So REM is not stored as the ASCII codes for R, E, and M. Instead, the entire word is converted to the value $B2. (The $ indicates the value is in hexadecimal notation. $B2 is equivalent to 178 in ordinary decimal notation.)

Another trick is using the character that indicates the end of a program line. You would assume (because you hit RETURN to tell the computer you have finished entering a line) that it would store the ASCII code for RETURN, $0D (13). But it doesn't. Instead, it stores $00 (0).

A third trick is the conversion of all line numbers to two bytes. A line number of 1 is stored as $01 00, and a line number of 256 is stored as $00 01. The high-order (more significant) byte is in the second position.

The machine language program puts this information to good use. Every time it encounters $00, it skips over the line number (and two more bytes which hold the location of the next line) to the beginning of the next command sequence. If it finds a value of $B2, the token for REM, when it is looking for REM statements, it jumps to the subroutine that switches one character for another. If the subroutine encounters a $00, or the ASCII token for ":", it ends and the program starts looking for the next REM statement.

Here's a list of some tokens and ASCII values of interest. You can find a list of ASCII codes used by Applesoft on pages 138 and 139 of the *Applesoft BASIC Programming Manual*. The tokens for the commands can be found on page 121.

Hex	Decimal	Printed As	
$B2	178	REM	(Token)
BA	186	PRINT	(Token)
84	132	INPUT	(Token)
8B	139	IN#	(Token)
8A	138	PR#	(Token)
23	35	#	(ASCII)
01	1	(Control-A)	(ASCII)
0D	13	(RETURN)	(ASCII)

You should know that DOS commands in a BASIC program are not tokenized. In

```
10 PRINT CHR$(4);"PR# 1"
```

PR# is stored as the ASCII equivalents for P, R, and #. Take this into consideration when setting up the machine language program. The token to search for in such a situation is $23, the ASCII code for #.

Changing Switch Without Loader

Let's call the BASIC program listed with this article *Loader* and the machine language program that it produces *Switch*. Once you have run Loader, you can change Switch, without rerunning Loader, using POKE commands.

To change the command token, use POKE 796, (new token).

To change the byte to be replaced, use POKE 815, (new byte).

To change the replacement, use POKE 821, (new byte).

Here's an example. If you want to change all of the control-B's in all of your PRINT statements to control-G's (bell ringers), you must first know that the token for PRINT is 186, that the ASCII byte for control-B is 2, and that the ASCII byte for the bell character is 7. Then enter:

```
] 10 POKE 796,186 : POKE 815,2 : POKE 821,7
```

The equivalent monitor command line is:

```
* 31C:BA N 32F:02 N 335:07
```

(The N allows you to put more than one command on a line.) Then enter & to make the change (or 300G in machine language).

Some Quick Facts About The Program

The machine language program can be placed anywhere in memory. Normally it resides at $300-$350 (768 to 848).

Locations $F9 and $FA (249 and 250) are normally unused by BASIC, DOS, or the monitor, but are used by Switch to keep track of its current point in the BASIC program it is changing.

Switch gets its information for the beginning and end locations of the program from $67 and $68 (103 and 104) and $AF and $B0 (175 and 176), respectively.

The & vector must be set to $300 (768). This is done by Loader.

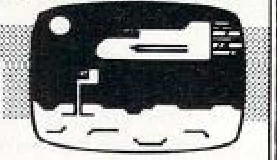
Bytechanger

```
10  REM


SWITCH LOADER

20  HOME : REM  CLEAR SCREEN
30  PRINT "THIS UTILITY WILL ALLOW YOU
     TO MAKE"
40  PRINT "GLOBAL CHANGES IN YOUR PROGR
     AM.  IT IS"
50  PRINT "SET UP TO CHANGE ALL CTRL-A'
     S IN REM "
60  PRINT "STATEMENTS TO RETURNS."
70  RESTORE : GOSUB 320: REM POKESWITCH
      INTO MEMORY
80  PRINT : INPUT "WOULD YOU LIKE TO CH
     ANGE IT? Y/N ";A$
90  IF  LEFT$ (A$,1) = "N" THEN 280
100  IF  LEFT$ (A$,1) <  > "Y" THEN 20
110  HOME
120  VTAB 10: PRINT "I WANT TO CONVERT
     THIS CHARACTER: ";: GET A$: PRINT
     A$: REM  'GET'
     ALLOWS YOU TO GRAB CARRIAGE
     RETURNS AND ESCAPES
130  PRINT "TO THIS CHARACTER: ";: GET
     B$: PRINT B$
140  PRINT "IN ALL"
150  PRINT "   1 REM"
160  PRINT "   2 PRINT"
170  PRINT "   3 INPUT"
180  PRINT "   4 IN#   (NO DOS)"
190  PRINT "   5 PR#   (NO DOS)"
200  PRINT "   6 #     (DOS IN USE)"
210  PRINT "STATEMENTS.  CHOOSE # ";: GET
     C$: PRINT C$
220  IF  VAL (C$) = 2 THEN  POKE 796,18
     6
230  IF  VAL (C$) = 3 THEN  POKE 796,13
     2
240  IF  VAL (C$) = 4 THEN  POKE 796,13
     9
250  IF  VAL (C$) = 5 THEN  POKE 796,13
     8
260  IF  VAL (C$) = 6 THEN  POKE 796,35
270  POKE 815, ASC (A$): POKE 821, ASC
     (B$): REM  POKE ASCII
     VALUES INTO SWITCH
280  POKE 1013,76: POKE 1014,0: POKE 10
     15,3: REM  POKE IN THE
     VECTOR FOR THE '&' COMMAND
290  PRINT : PRINT "USE '&' TO CONVERT"
300  END
310  REM
```

```
DATA FOR SWITCH

320   FOR A = 768 TO 848: READ B: POKE A
      ,B: NEXT : RETURN
330   DATA 169,0,133,249,165,104,133,250
      ,169,3,24,101,103,144,2,230,250,16
      8,200
340   DATA 208,2,230,250,177,249,240,31,
      169,178,209,249,208,241,200,208,2,
      230,250
350   DATA 177,249,240,16,201,58,240,228
      ,169,1,209,249,208,237,169,13,145,
      249,208
360   DATA 231,162,4,200,208,2,230,250,2
      02,208,248,165,250,197,176,144,200
      ,240,198
370   DATA 196,175,144,194,96
380   REM
      THIS REMARK WAS PRECEDED
      BY A CARRIAGE RETURN AND
      FIVE SPACES ON EACH LINE
```
©

SOFTWARE FOR THE COMMODORE 64: PICTURE PERFECT © MW 323—$24.95

Feature Highlights: High resolution drawing routine draws circles, lines, boxes, graphs & pictures easily; freestyle drawing with joystick; five different pens; upper and lower case characters; save screens to disk; 15 colors available; three demo screens included.



COMMODORE 64 SOFTWARE SPECIALS

Totl Text 2.6 (tape)	reg $40.00	NOW $29.95
Totl Text 2.6 (Disk)	reg $44.00	NOW $32.95
Totl Tabl 2.6 (tape)	reg $20.00	NOW $14.95
Totl Tabl 2.6 (disk)	reg $24.00	NOW $17.95
Totl Time Manager 2.6 (tape)	reg $35.00	NOW $26.25
Totl Time Manager 2.6 (disk)	reg $39.00	NOW $29.25
Research Asst 2.0 (tape)	reg $35.00	NOW $26.25
Comm-Data (tape)	reg $19.95	NOW $14.95

TITLES: Pacacuda, Logger, Escape MCP, Centopods, Ape Craze.

VIC Bonus 60 Programs MW 324 $19.95

Computermat Arcade Pak (3 games) reg $24.95 NOW $17.49
Computermat Super Pak (10 games) reg $59.95 NOW $44.95

Micro World Electronix, Inc.
3333 S. Wadsworth Blvd #C105,
Lakewood, CO 80227
(303) 934-1973 or (303) 987-2671

VISA MasterCard

Commodore 64

FIRE ANTS & KILLER BEES

- A 100% Machine Language Arcade Style Action Game
- Two Player
- Multi Level
- Developed On The 64 For The 64

Disc $24.95
Cassette $22.95

Fight off columns of ARMY ANTS... swarms of KILLER BEES... the angry QUEEN BEE and the menacing QUEEN FIRE ANT!! OH!, Don't forget the GHOST ANTS!!

Send Check or Money Order
Personal checks accepted
Allow 3 weeks to clear

WizardWare
P.O. Box 1046 — Dept. C8
Bolingbrook, Illinois 60439

Warranted
Dealer Inquiries Invited

DEVELOP-64

FIVE POWERFUL SOFTWARE DEVELOPMENT TOOLS

Plus The Exciting New Book

INSIDE THE COMMODORE 64™

THE BOOK

A complete clear explanation of machine language, Assembly language, Commodore 64 architecture, graphics, joystick and sound effect programming. Detailed step-by-step guide to the use of the development tools. How to combine BASIC and machine language, make auto-start cartridges, interface with the internal ROM-based programs of BASIC and the Kernal. Sample programs fully explained.

THE TOOLS

Assembler/Editor/Loader/Decoder/Monitor
Full-featured Assembler allows use of labels, comments and arithmetic expressions to create machine language programs. Create, save, modify Assembly language programs with the Editor. Load and link machine language modules with the Loader. Decode machine language back into assembly language for study or input to the Editor. Single-step program execution with the Monitor. Combines Assembler/Editor for maximum ease of use.

ALL FOR $54.95 PLUS $2.00 POSTAGE AND HANDLING Add $5.00 for disk version.

Send check, M.O., VISA/MC ($2.00 S.C.) or specify C.O.D. (add $3.00) to:

French Silk smooth ware

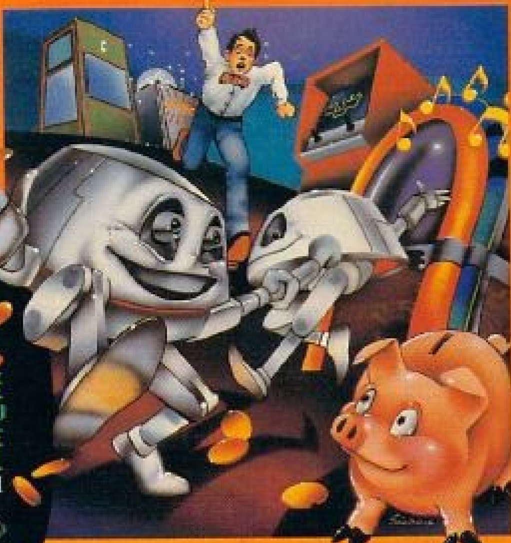
P.O. Box 207, Cannon Falls, MN 55009

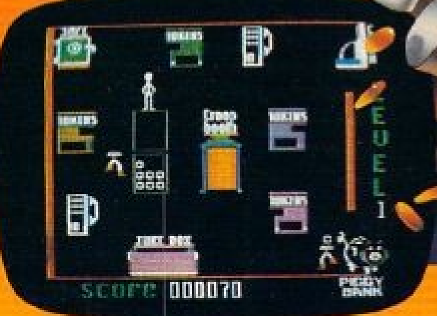
507-263-4821

Commodore 64™ is a registered TM of Commodore Business Machines Inc.

A million laughs

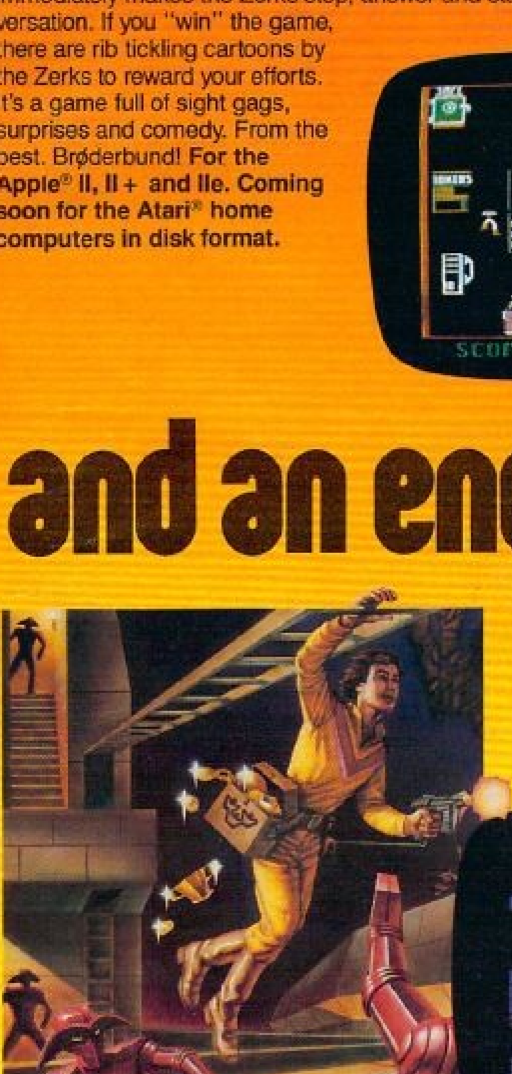
SPARE CHANGE™ You are the game-happy owner of the Spare Change Arcade. Two fun-loving, but overworked Zerks—the main characters in your most popular game—have broken loose and are trying to retire from the business. You try madly to stop them. If you can get a coin into the juke box, the Zerks get so caught up in the music, they drop everything and start dancing. You also try popping popcorn and making a pay phone ring—which immediately makes the Zerks stop, answer and start a wild conversation. If you "win" the game, there are rib tickling cartoons by the Zerks to reward your efforts. It's a game full of sight gags, surprises and comedy. From the best. Brøderbund! **For the Apple® II, II+ and IIe. Coming soon for the Atari® home computers in disk format.**

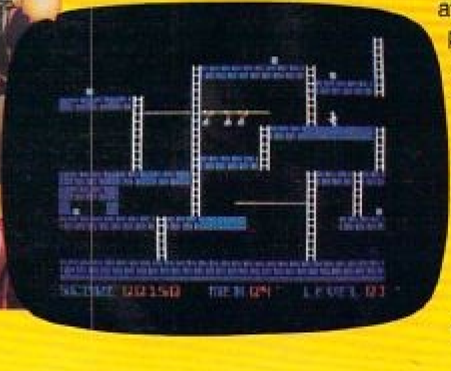




and an endless challenge

LODE RUNNER™ Here's a game that will never stop challenging you. That's because Lode Runner is more than a spellbinding, fast-action game with its 150 different mind-boggling game screens. Lode Runner is also an easy-to-use Game Generator that lets you create your own games. Without any knowledge of programming, you can easily design unique Lode Runnner screens, then bring them to action-packed life. You will maneuver through scene after scene, running, jumping, drilling passages and outfoxing enemy guards in a secret underground hideaway as you pick up chests of gold stolen from citizens of the Bungeling Empire. There's no end to the thrills, chills and challenge. Of course, it's from Brøderbund! **For the Apple II, II+ and IIe. Coming soon for the: Atari home computers (disk and cartridge); Atari 5200™ Super System; Commodore 64™; VIC-20™; IBM® PC.**

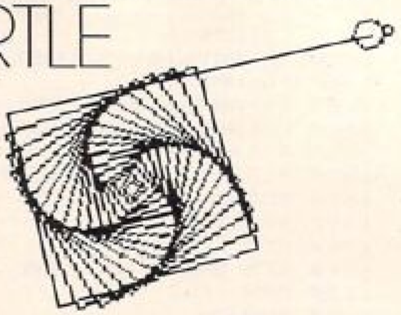




Ask your Brøderbund dealer for sneak previews.

Brøderbund Software **Discover the Difference**

1938 Fourth Street San Rafael, CA 94901



David D. Thornburg, Associate Editor

The Logo Kaleidoscope

One of the first programming projects for many BASIC programmers is the construction of a screen kaleidoscope that generates pretty, symmetrical patterns on the display screen. For these programs, people usually pick a screen location at random and then place a colored dot at that location and at three other "mirror" locations to produce four symmetrically placed dots. While the resulting image is often quite attractive, the result is not that of a true kaleidoscope.

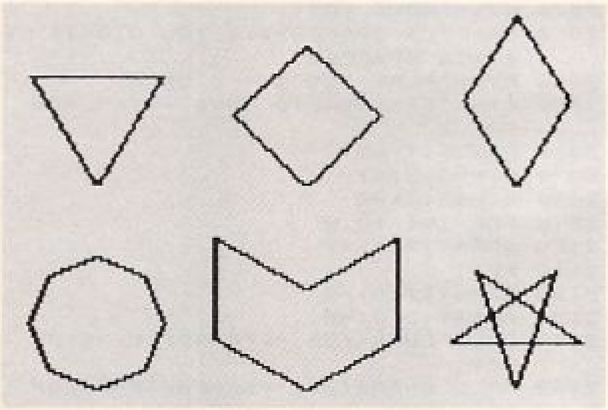
If you have ever taken a kaleidoscope apart, you must have wondered how such a simple apparatus could generate such beautiful images. Most kaleidoscopes consist of a set of mirrors and some small pieces of colored plastic that can be shaken to take random positions on a flat surface. When you look through the eyepiece, the mirrors generate multiple images of the arrangement of plastic pieces to produce beautifully symmetric pictures. Because Logo's turtle graphics allows you to easily create images that imitate the pieces of plastic, it is possible to create quite attractive kaleidoscopic images on your computer screen with a simple set of procedures.

The Logo kaleidoscope operates in the following manner. The system contains a set of graphic procedures to draw the fundamental picture elements (squares, triangles, stars, etc.). There can be as many of these elements as you desire (subject to the memory limitations of your system, of course). Each of these elements can be drawn as large as you desire. This gives the effect of having even more patterns to choose from.

Next, we use Logo's random number generator to select a shape, a size for the shape, the shape's color, and a distance from the center of the screen at which the shape will be drawn. Finally, this data is used by another procedure that places a copy of the chosen shape at several equally spaced angles around the center of the screen. Once one shape has been drawn, the process can be repeated for other shapes until the final image meets with your approval.

The kaleidoscope we will demonstrate in this article is written in the MIT version of Logo for the Apple II and should work with most Logo systems with very few modifications.

The kaleidoscope was started out with six shapes.



The procedures for these shapes are:

```
TO TRI :SIZE
  LT 30
  REPEAT 3 [FD :SIZE RT 120]
  RT 30
END
TO DIAMOND :SIZE
  LT 45
  REPEAT 4 [FD :SIZE RT 90]
  RT 45
END
TO PATT1 :SIZE
  LT 30
  REPEAT 2 [FD :SIZE RT 60 FD :SIZE RT 120]
  RT 30
END
TO OCT :SIZE
  LT 67.5
  REPEAT 8 [FD :SIZE / 2 RT 45]
  RT 67.5
END
TO PATT2 :SIZE
  LT 60
  FD :SIZE RT 60 FD :SIZE RT 120
  FD :SIZE LT 60 FD :SIZE RT 120
  FD :SIZE RT 60 FD :SIZE RT 120
END
TO STAR :SIZE
  LT 18
  REPEAT 5 [FD :SIZE RT 144]
  RT 18
END
```

Each of these figures has been defined to have mirror symmetry on the vertical axis. This is not a requirement, and you may wish to experiment with other orientations. The octagon was drawn at half the specified size to keep it in balance with the other figures.

Constructing The Pattern

To make the kaleidoscopic image, we need a procedure that creates a list of basic patterns, chooses a pattern at random from this list, and selects an appropriate size (say between 20 and 50 units). Next, it should pick a random distance from the center (less than 60 units, to keep the images on the screen). Once these steps have been completed, copies of the chosen image should be stamped symmetrically around the screen. Then the procedure should wait for you to tell it if you want another element added to the image. When you press the RETURN key, the process will be repeated. The following procedure performs these tasks for us:

```
TO IMAGE
  MAKE "LIST [STAR DIAMOND OCT PATT1
    PATT2 TRI]
  MAKE "NAME SENTENCE PICKRANDOM :LIST
    ( 20 + RANDOM 30 )
  MAKE "DIST RANDOM 60
  PENCOLOR ( 1 + RANDOM 5 )
  PENUP
  WINDMILL :DIST :NAME
  MAKE "NAME REQUEST
  IMAGE
END
```

This procedure uses two other procedures that have to be defined: PICKRANDOM and WINDMILL. The function of PICKRANDOM is to choose an element of a list randomly. The following procedure does this for us:

```
TO PICKRANDOM :LIST
  OUTPUT PICK ( 1 + RANDOM (LENGTH :LIST ) )
    :LIST
END
```

The procedure PICK selects a given element from a list, and LENGTH measures the number of elements in a list:

```
TO PICK :NUM :LIST
  IF :NUM = 1 OUTPUT FIRST :LIST
  OUTPUT PICK ( :NUM - 1 ) ( BUTFIRST :LIST )
END
TO LENGTH :LIST
  IF :LIST = [] THEN OUTPUT 0
  OUTPUT 1 + LENGTH BUTFIRST :LIST
END
```

These two procedures operate "recursively." If you have a hard time understanding how they work, you may want to read about them in *Logo for the Apple II*, by H. Abelson, or read the chapter on recursion in my book *Discovering Apple Logo*. Also, we published some columns on recursion in "Friends of the Turtle" (COMPUTE!, November and December 1982).

Defining Windmill

The only procedure we have left to define is WINDMILL. The function of this procedure is to draw a chosen pattern at equally spaced angular increments around the center of the screen. You may want to experiment with different numbers of images. I have tried using six images spaced at 60-degree increments and eight images spaced at 45-degree increments. These both work fine, but other angles are worth exploring as well. The number of copies of a pattern times the angle increment must be 360 in order for the pattern to be symmetric. That is why we turn 60 degrees for 6 copies (6 x 60 = 360) and 45 degrees for 8 copies (8 x 45 = 360).

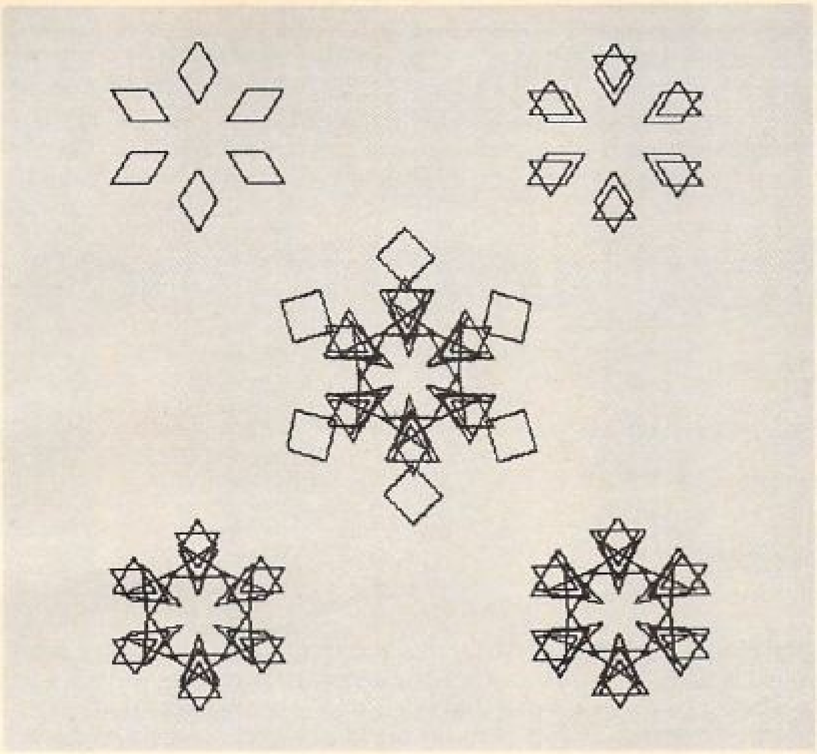
```
TO WINDMILL :DIST :LIST
  REPEAT 6 [FD :DIST PENDOWN RUN :LIST
    PENUP BACK :DIST RT 60]
END
```

To generate a kaleidoscopic pattern, hide the turtle and enter:

```
IMAGE
```

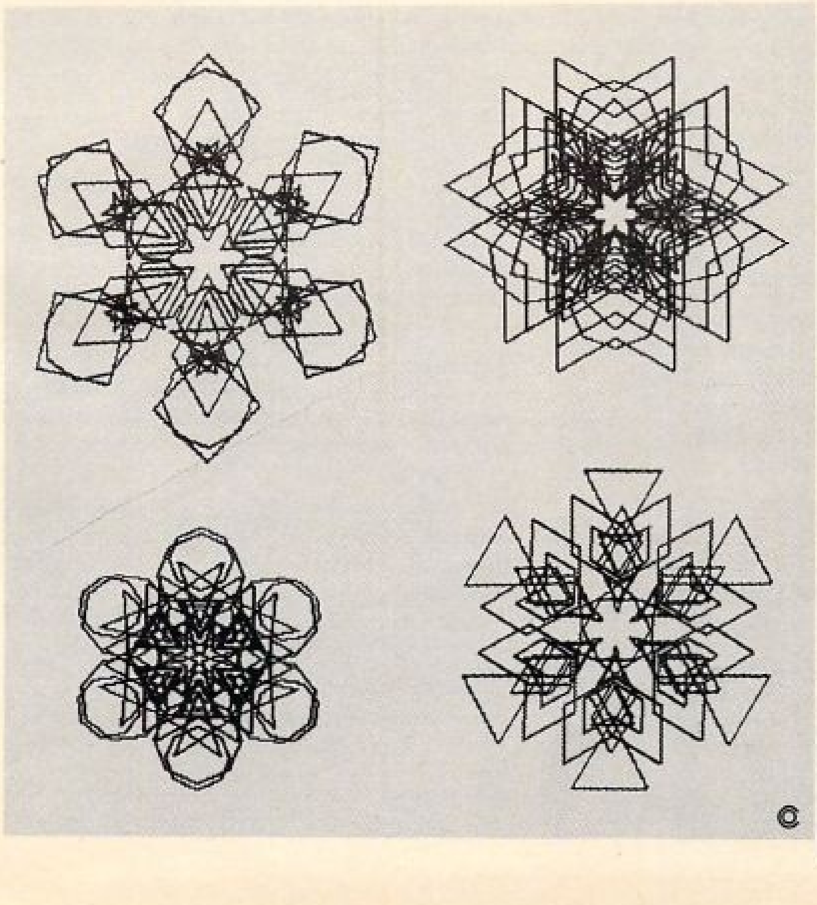
After the first pattern is drawn, press RETURN to get the next one. When the complexity of the pattern is satisfactory, you may want to print a copy of it or save it on your disk (with SAVEPICT, for example). If you are ambitious, you might want to write a Logo procedure that will keep track of all the randomly chosen values and generate its own Logo procedures for each pattern. Abelson's book (mentioned above) shows how to do this sort of thing.

The following five pictures show the successive development of one pattern:

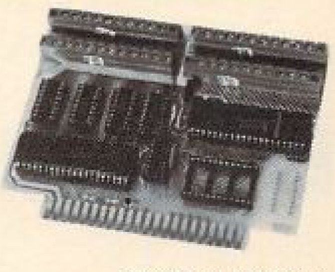


The remaining figures illustrate some other kaleidoscopic patterns that were generated with this set of procedures.

I think you will agree that these patterns are more interesting than those created with colored dots.



NEW!
Universal Input/Output Board for VIC-20/64

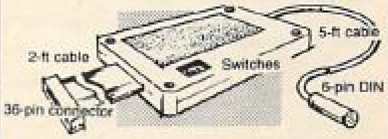


16-channel A/D converter & 1-channel D/A output; 16 high-current discrete outputs.

• University physics & electrical engineering labs and hospital applications.

VIC-20 uses MW-311V **$185.00**
CBM-64 uses MW-311C **$225.00**

MW-302: VIC-20/64 Parallel Printer Interface.



Works with all centronics type parallel matrix & letter printers and plotters—Epson, C.Itoh, Okidata, Nec, Gemini 10, TP-I Smith Corona, and most others. Hardware driven; works off the serial port. Quality construction: Steel DIN connectors & shielded cables. Has these switch selectable options: Device 4, 5, 6 or 7; ASCII or PET ASCII; 7-bit or 8-bit output; upper & lower case or upper only. Recommended by PROFESSIONAL SOFTWARE for WordPro 3 Plus for the 64, and by City Software for PaperClip.

MW-302 .... **$119.95**

Dealer inquiries invited.

Micro World Electronix, Inc.
3333 S. Wadsworth Blvd. #C105,
Lakewood, CO 80227
(303) 987-2671

ISAM

Building Your Own Random File Manager

Michael D. Lipay

There are several approaches to handling computer files (collections of data). Among the fastest and best is the random access disk file which uses special techniques to quickly locate any piece of information from anywhere within the entire file.

This tutorial explains how random access can be achieved and examines alternative ways to process data files. It includes a sample program, written in Applesoft BASIC, but which can easily be adapted to work on other computers using Microsoft BASIC.

Besides protecting earth from aliens, a main purpose of a computer is processing information. This data processing can be anything from keeping track of your stamp collection to maintaining a running inventory for your business. When it becomes necessary to retain the information long after the computer has been turned off, tape or disk storage is used.

Magnetic storage devices are capable of storing information indefinitely (provided they are kept clean and away from magnetic fields). Basically, there are two types of magnetic storage devices available to the micro computer user – tape and disk. Both devices are capable of storing large amounts of information, and do so in groups called files. A file is a collection of related information, and the user has three primary types of files to select from:

I) Sequential Tape Files
II) Sequential Disk Files
III) Random Access Disk Files

Which of the three you decide to use for a given program will depend on many factors. Each has its own advantages and disadvantages; they are discussed here in an effort to help you select the best one for your needs.

Sequential Tape Files

If you have large amounts of data which you do not need to process frequently, then tape files should be considered. Tapes can store vast amounts of data in a relatively compact space, and at a very low price. Tapes serve as an excellent medium to keep a backup of disk programs and files. The big drawback to using tapes is that they are slow, so make sure you have plenty of time.

Sequential Disk Files

Sequential disk files are best if you have small amounts of data to process. The files have the advantages of being faster than tape and more space conservative than random access files. Probably the only disadvantage of sequential disk files is the slowness of updating large files. In order to change a single record on a sequential file, you must copy all records to a work file, changing any records desired along the way, then delete the old file and rename the work file. This could be as time consuming as tape files, were it not for the speed of the disk.

Random Access Disk Files

Large volumes of data which must be updated with any frequency should be held in random access files. This type of file lets you easily update any given record without having to process or read through any other record on the file. It also has disadvantages such as requiring all records to be of the same, fixed length and needing to know where on the file a particular record is located.

There are several methods available to help you to determine where a particular record is located on a random access file. John Hudson covered the HASH/LINK method in the March 1982 issue of COMPUTE!. He did an excellent job; and if you desire to learn more about it, I suggest that you read this article. The HASH/LINK method does have some problems. For example:

I) If you fill the overflow area, you will have to reorganize the file again.

II) As soon as you initialize the random file, you take up more space than you may need.

III) Successive "collisions" can greatly increase access time (rec 100 links to rec 212, rec 212 links to rec 487, rec 487 links to...).

IV) Expanding the main and overflow areas of the file may require major program revisions (deciding the main area should be 2000 recs instead of 1000 recs will require changes to your hashing logic), as well as requiring you to reload the file.

V) Sequential (ascending or descending) processing is almost impossible.

VI) If you need to "key" on an alphabetic field (such as a name), you must first convert it to a numeric value.

VII) Once the file has been created, it is impossible to select an alternate key (e.g., a file is hashed on the last name, but you need a report in social security number order).

VIII) Deleting a record requires several Read/Write steps to keep the link field updated. Once a record has been deleted, the position that it occupied on the file is unusable, since all adds occur at the end of the file.

In the rest of this article I will cover an alternate method known as Indexed-Sequential Access Method (ISAM).

ISAM

ISAM can solve all the problems associated with HASH/LINK files, but it has some problems of its own. ISAM works on the principle that it is faster to search memory than a disk. Unfortunately, before you can search memory, you must have something in it, and this is the problem with ISAM.

ISAM works by loading the desired "key" field of each record in a file into an array. This is done by placing the key field of the first record into the first position of the array, the key from the second record into the second position of the array, etc. Once the array has been loaded, you simply search the array for the desired key; its position in the array is the record number for the random access file. Described below are the procedures necessary for the most common types of file processing:

I) ADD A RECORD
- a) Search the array to determine if the record already exists.
- b) Move the new "key" to the end of the array, or to the first "open" position in the array.
- c) Use the position number of the array to write the record to the file.

II) DELETE A RECORD
- a) Find the key in the array.
- b) "Open" the entry in the array by moving a "dummy" key into the array (such as zeros).
- c) Write the dummy values to the file.

III) CHANGE A RECORD
- a) Find the key in the array.
- b) Use the position number to read in the record.
- c) Make your change to the record (even change the key).
- d) Write the new record to the file using the position number.
- e) If you changed the key, move the new key into the array.

IV) PROCESS SEQUENTIALLY BY KEY
- a) Sort the array into the desired order (ascending or descending).
- b) Process the records sequentially through the array.

V) PROCESS BY A DIFFERENT KEY
- a) Load the array with the new keys from the file.
- b) Process normally using the new array.

Listed below are sample programs, written in Applesoft, which illustrate ISAM programming techniques. The programs are shells which can easily be modified to suit your own purposes. Note that all branch instructions bypass the REM statements; thus, if you want to key the program in without remarks, no line numbers will have to be changed. Variables used in the programs are:

D$ – Control-D (disk access)

IA – Index Array

IE – Index End (last entry used)

IP – Index Pointer (entry number for the part searched for)

IO – Index Open (entry number for first "open" or empty record)

FOUND – Switch to indicate if part searched for is in the index:

0 – part not in index
1 – part in index

PART – Part number being searched for

10-13 This section goes to a one-time routine to load the index array with the desired key field (in this case a part number).

100-114 Display the options available in a menu format.

120-122 This gets the option into a string. Then, using the VAL command, goes to the appropriate routine. Note that if zero, a non-numeric character, or a number greater than five is entered, the menu is displayed again.

200-215 The index array is searched sequentially in this section. If the key is found, the following values are returned:

```
FOUND =1
IP    =Entry in array for desired key
IO    =First open entry in array (entry
       with key of zero)
```

If the key is not found, the following values are returned:

```
FOUND =0
IO    =First open entry in array
```

Note on lines 212 and 213 the method used to exit from the FOR/NEXT loop. This is the method suggested by Apple to exit the loop from other than normal completion. Its purpose is to prevent ?OUT OF MEMORY errors from occurring as a result of too many "open" loops.

300-324 ADD A PART

310 Accepts the part number to be added to the file.

311 Goes to the routine to search the index. If the part already exists (FOUND=1), an error message is displayed and control is returned to the menu.

321-322 The new part is written to the master file using the open entry pointer (IO) as the record number.

323 If the new part is added to the end of the file, the number of the last entry (IE) is updated.

324 Returns to the menu.

400-424 DELETE A PART

410 Accepts the part to be deleted.

411 Goes to the search routine. If the part is not on file (FOUND=0), an error message is displayed and control is transferred to the menu.

420 The part is removed from the index by making the entry zero.

421-422 The part is removed from the master file.

423 If the part was the last one in the array, the ending pointer (IE) is reduced by one.

424 Return to the menu.

800-813 UPDATE INDEX POINTER

810-811 Write the number of the last entry in the index to record zero of the master file.

812 Closes the master file.

813 Stops the program.

900-930 LOAD THE INDEX ARRAY

910 Initially sets up variables.

911 Sets up an error routine to handle end-of-data and not-found conditions.

912 Opens the master file.

913-914 Read the number of the last record on the master file.

915 Turns off the error routine, dimensions the index array to allow up to ten records to be added to the end of the array (this can be changed to allow for more expansion).

916 If no records exist on the master, control goes to the menu.

920 Sets up the error routine.

921-924 Load the key field (part number) into the array.

930 Turns the error routine off; returns to the menu.

The second program offers a different method of handling the index. Type in lines 10-630 from Program 1, then add the lines from Program 2. In this program the index is kept on a sequential disk file, for speed of loading the array.

800-833 Save the index array.

810 Check the index change switch; if it is zero, the index has not changed and does not have to be rewritten. Control goes to 832.

811 Deletes the index file.

820-823 Write the array to the index file.

830-831 Write the number of the last entry in the index to record zero of the master file.

832 Closes the master file.

833 Stops the program.

900-940 LOAD THE INDEX ARRAY

910 Initially sets up variables.

920 Opens the master file.

921 Sets up the error routine.

922-923 Read the number of entries in the index file.

930 Sets up a new error routine.

931 Dimensions the index array (with expansion of 10).

932-934 Read the index file into the array.

935 Turns the error routine off and closes the index.

940 Turns control over to the menu.

Program 1: ISAM

```
10  REM
11  REM  CALL INDEX LOAD ROUTINE
12  REM
13  GOTO 910
100  REM
101  REM  SELECT OPTION
```

```
102  REM
110  HOME : PRINT "1) ADD PART"
111  PRINT "2) DELETE PART"
112  PRINT "3) CHANGE PART"
113  PRINT "4) DISPLAY PART"
114  PRINT "5) STOP"
120  PRINT : INPUT "SELECT OPTION: ";OPT$
121  ON  VAL (OPT$) + 1 GOTO 110,310,410,510
     ,610,810
122  GOTO 110
200  REM
201  REM  SEARCH INDEX ARRAY
202  REM
210 IO = IE + 1: IF IE = 0 THEN FOUND = 0:
     RETURN
211  FOR I = 1 TO IE
212  IF IA(I) = PART THEN IP = I:I = IE + 1:
      NEXT :FOUND = 1: RETURN
213  IF IA(I) = 0 AND IO = IE + 1 THEN IO =
     I: NEXT
214  NEXT I
215 FOUND = 0: RETURN
300  REM
301  REM  ADD A PART
302  REM
310  INPUT "ENTER NEW PART NUMBER: ";PART
311  GOSUB 210: IF FOUND = 1 THEN  PRINT "PA
     RT ALREADY ON FILE": GOTO 110
320 IA(IO) = PART
321  PRINT D$;"WRITE MASTER,R";IO
322  PRINT PART: PRINT D$
323  IF IO > IE THEN IE = IO
324  GOTO 110
400  REM
401  REM  DELETE A PART
402  REM
410  INPUT "ENTER PART TO BE DELETED: ";PART

411  GOSUB 210: IF FOUND = 0 THEN  PRINT "PA
     RT IS NOT ON FILE": GOTO 110
420 IA(IP) = 0
421  PRINT D$;"WRITE MASTER,R";IP
422  PRINT 0: PRINT D$
423  IF IP = IE THEN IE = IE - 1
424  GOTO 110
500  REM
501  REM  CHANGE A PART
502  REM
510  INPUT "ENTER PART TO BE CHANGED: ";PART

511  GOSUB 210: IF FOUND = 0 THEN  PRINT "PA
     RT IS NOT ON FILE": GOTO 110
520  PRINT D$;"READ MASTER,R";IP
521  INPUT PART: PRINT D$
530  REM  CODING TO CHANGE PART
540 IA(IP) = PART
541  PRINT D$;"WRITE MASTER,R";IP
542  PRINT PART: PRINT D$
543  GOTO 110
600  REM
601  REM  DISPLAY PART
602  REM
610  INPUT "ENTER PART NUMBER: ";PART
611  GOSUB 210: IF FOUND = 0 THEN  PRINT "PA
     RT IS NOT ON FILE": GOTO 110
612  PRINT D$;"READ MASTER,R";IP
613  INPUT PART: PRINT D$
620  REM  CODING TO DISPLAY PART
630  GOTO 110
800  REM
801  REM  UPDATE INDEX POINTER
802  REM
810  PRINT D$;"WRITE MASTER,R0"
811  PRINT IE
812  PRINT D$;"CLOSE MASTER"
813  END
900  REM
901  REM  LOAD INDEX ARRAY
902  REM
910 D$ =  CHR$ (4):IE = 0:IP = 0:IO = 0:FOUN
     D = 0:PART = 0
911  ONERR  GOTO 915
912  PRINT D$;"OPEN MASTER,L25"
913  PRINT D$;"READ MASTER,R0"
914  INPUT IE: PRINT D$
915  POKE 216,0: DIM IA(IE + 10)
916  IF IE = 0 GOTO 110
920  ONERR  GOTO 924
921  FOR I = 1 TO IE
922  PRINT D$;"READ MASTER,R";I
923  INPUT IA(I)
924  NEXT I: PRINT D$
930  POKE 216,0: GOTO 110
```

Program 2: Index Array Routine

```
800  REM
801  REM  SAVE INDEX
802  REM
810  IF IC = 0 GOTO 832
811  PRINT D$;"DELETE INDEX"
820  PRINT D$;"OPEN INDEX"
821  PRINT D$;"WRITE INDEX"
822  FOR I = 1 TO IE: PRINT IA(I): NEXT I
823  PRINT D$;"CLOSE INDEX"
830  PRINT D$;"WRITE MASTER,R0"
831  PRINT IE
832  PRINT D$;"CLOSE MASTER"
833  END
900  REM
901  REM  LOAD INDEX ARRAY
902  REM
910 D$ =  CHR$ (4):IE = 0:IP = 0:IC = 0:IO =
     0:FOUND = 0:PART = 0
920  PRINT D$;"OPEN MASTER,L25"
921  ONERR  GOTO 930
922  PRINT D$;"READ MASTER,R0"
923  INPUT IE
930  ONERR  GOTO 935
931  DIM IA(IE + 10)
932  PRINT D$;"OPEN INDEX"
933  PRINT D$;"READ INDEX"
934  FOR I = 1 TO IE: INPUT IA(I): NEXT I
935  POKE 216,0: PRINT D$;"CLOSE INDEX"
940  GOTO 110
```

©

ADVERTISEMENT

Educational News

Orange Plus Educational News located at 23801 Calabasas Road/Suite 2050/Calabasas, CA 91302/(213) 999-5210

Legal Apple^{Type} Compatible

New Computer runs CP/M & Apple^{Type} Software

CALABASAS — Now instructors can teach Apple-type programs such as LOGO in one class and professional-style word processing in another without any hardware changes. Dual microprocessors (Z80-A & 6502, two computers in one case), with the radical new Orange Plus development the "EuroROM", allows the machine to read/write/work with Apple-type software as well as CP/M programs and access either CPU via the keyboard.

The ***"ORANGE+TWO"***™ is a brand new direction in the evolution of the personal computer. ORANGEFORTH-83, a Fig-FORTH derivative and readily accessible public domain language, is resident in the ROM*. Also included is CP/M 3.0, Digital Research's latest CP/M version. There is also a built-in disk drive controller for two Apple-type drives, a cassette interface, joystick port, color graphics and ASCII keyboard with numeric keypad. For expert word processing, the keyboard features upper and lower case (lower case characters are true descenders) with auto repeat.

This new breed of computer is a breakthrough for educators and school systems throughout the world. The ability to run both Apple-type and CP/M software on the same machine relieves financially-pressed educators from expensive equipment burdens, allowing them to spend more money where it counts...**on the teachers.**

*Available on disk at extria cost.

MORE FOR LESS

The ***"ORANGE+TWO"***™ 64K computer includes these standard features:

Built-in CP/M 2.2 compatibility	**no charge**
Digital Research's CP/M 3.0	**no charge**
Built-in disk drive controller for two Apple-type drives	**no charge**
Numeric keypad with separate return key	**no charge**
110/220 volt switch selectable power supply, 50-60 Hz	**no charge**
Full function ASCII keyboard with auto repeat	**no charge**
Cassette interface	**no charge**
Fully grounded metal base plate	**no charge**
Adjustable audio volume control	**no charge**
ORANGEFORTH-83, Z80-A FORTH language, resident in ROM	**no charge**
Z80-A CPU, a second computer	**no charge**
6 slot double sided logic board	**no charge**
Bank switchable RAM, fully socketed, expandable to 256K	**no charge**
Programmable 2764 EPROMs	**no charge**
TOTAL: ***"ORANGE+TWO"***™	**$1095.00**

Compare these features included on the ***"ORANGE+TWO"***™ *with the competition's over $2000 for equivalent product.*

THE LEASING ALTERNATIVE

CALABASAS — Through select leasing companies, Orange Plus Computer Systems will offer the following lease programs to qualified corporations and educational institutions: An ***"ORANGE+TWO"***™ computer with a green monitor and one disk drive for only $59.55 per month...36 month closed end lease. Alro included is a full maintenance program and all revisions and upgrades that may be available during the term of the lease. Subject to credit approval. Call for additional information.

DISTRIBUTORS: CALL (213) 999-5210

CALABASAS — Due to the tremendous amount of individual inquiries and dealer applications, qualified stocking distributors are needed. Select areas available!

MORE PRODUCTS FROM ORANGE PLUS COMPUTER SYSTEMS TO ENHANCE YOUR *"ORANGE+TWO"*™, APPLE II/II PLUS, OR FRANKLIN ACE 1000

Disk-based ORANGEFORTH-83 language with full documentation	**$99.95**
Digital Research's™ CBASIC with disk and documentation	**$49.95**
Orange Plus KoalaPad™ Touch Tablet w/Micro Illustrator™	**$124.95**
Orange Plus Joystick (works on "ORANGE+TWO", Apple II, II+, & //e)	**$29.95**
Orange Plus self-centering joystick	**$39.95**
10MB (Formatted) 5¼" half-height Winchester Hard Disk Drive	**$1495.00**
Controller & Interface for Hard Disk	**$395.00**

Includes everything necessary to be installed in ***"Orange+Two"****, Apple or Franklin Computer*

The above are suggested retail prices. Prices may vary from state to state.

Collins International Trading Corporation,
23801 Calabasas Road, Suite 2050, Calabasas, CA. 91302

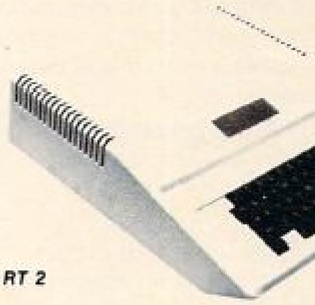
IT'S HERE AT LAST!

THE FIRST LOW COST 6502 COMPUTER WITH 48K RAM THAT'S COMPLETELY COMPATIBLE WITH APPLE II®

Introducing the RAM-TECH* *RT 2* computer which offers you greater flexibility than the Apple II+® at a price that's easier on your finances. The *RT 2* will run all software and use all peripheral hardware designed for the Apple II+.® It's 100% Apple® compatible yet it offers you features not available on the Apple II+.®

COMPARE THESE FEATURES:

— supports upper & lower case characters from the keyboard
— high quality ABS case will not crack and is light weight
— RF modulator included in addition to composite video output
— high quality light-touch keyboard with upper/lower case key
— 15 key numeric key pad for fast data entry (optional)
— high power switching power supply (8.5 amps total) will power two disk drives and several peripherals easily, has internal circuit breaker
— eight expansion slots to increase flexibility with peripheral cards
— 48K RAM expandable to 64K (192K RAM upgrade card available soon)
— socketed IC's for quick and easy service
— thousands of programs (business & home) available
— operates identically to the Apple II+® with exception of extra features on *RT 2*
— full 90 days parts and labor warranty with fast service turn-around



THE RAM-TECH* RT 2 IS EVERYTHING YOU ALWAYS WANTED FROM AN APPLE® (AND MORE), BUT THOUGHT YOU COULDN'T AFFORD. AND THAT'S GOOD NEWS!

BUT THE BEST NEWS IS THE PRICE:
$525.00 + $18.00 shipping & handling

WITH OPTIONAL NUMERIC KEYPAD:
$575.00 + $18.00 shipping & handling

COMPARED TO APPLE II® WHICH RETAILS AT MORE THAN DOUBLE THIS PRICE AND OFFERS YOU LESS, YOU CAN SEE WHY THE RAM-TECH* *RT 2* SHOULD BE YOUR CHOICE. *MAKE NO SYNTAX ABOUT IT, THE RT 2 IS FOR YOU!*

PERSONAL I.D. OPTION — We will burn your name into ROM. When you power up your Ram-Tech your name will be displayed on the screen. ***AN EXTRA MEASURE OF SECURITY AGAINST THEFT.***
Indicate on your order what name you wish to use (maximum 8 letters including spaces) — ADD: $25.00 for I.D. option.

ALSO AVAILABLE: (sub-assemblies for do-it yourselfers) — 90 days warranty

1 – ABS case, high quality, Apple® look-a-like and color matched, will replace Apple® case	$ 68.50
2 – same as #1 above but with numeric keypad cutout	$ 84.00
3 – keyboard with upper/lower case key, direct replacement for Apple® or Ram-Tech*	$110.00
4 – same as #3 above but with numeric keypad, use in combination with #2, also fits Apple®	$139.50
5 – RF modulator with TV switch and cable, also fits Apple®	$ 19.50
6 – switching power supply, 8.5 amps, internal circuit breaker, high power, also fits Apple®	$112.00
7 – motherboard, completely assembled & tested, fully operational with burned ROMS	$295.00
8 – bare motherboard w/ parts discription screened on board and instructions, highest quality G10 epoxy	$ 48.00

SHIPPING & HANDLING FOR ABOVE: cases #1 & 2 add 10%; all others add 5%

PERIPHERALS, CARDS & ACCESSORIES FOR YOUR RAM-TECH* OR APPLE®

all items #1 to 19 guaranteed for one year (disk drives for six months)

1 – 16K RAM card	$ 58.00	11 – interger card	$ 97.50
2 – 80 column card	$ 94.50	12 – PAL card	$114.00
3 – clock card	$135.00	13 – language card	$ 72.50
4 – communications card	$ 89.00	14 – RS-232 serial card	$ 84.00
5 – disk controller card	$ 58.00	15 – Z-80 CP/M card	$ 92.00
6 – EPROM writer card	$124.50	16 – VERSA card	$345.00
7 – FORTH card	$ 98.50	17 – analog/digital card	$116.50
8 – IEEE-488 card w/cable	$186.00	18 – system cooling fan	$ 63.00
9 – parallel printer card w/cable	$112.00	19 – joy stick w/button	$ 24.50
10 – parallel printer buffer and grappler card with cable			$245.00

DISK DRIVES: Guaranteed high quality for use with all Apple® compatible (or Ram-Tech*) controllers – DOS 3.3 & 3.2

USES SPECIAL LOW POWER CONSUMPTION CIRCUITRY

TEAC 55A	$325.00	$370.00 with controller
SHUGART	$285.00	$330.00 with controller

SHIPPING & HANDLING: add $3.50 per item #1 - 19 . . . disk drives add $10.00

TO ORDER

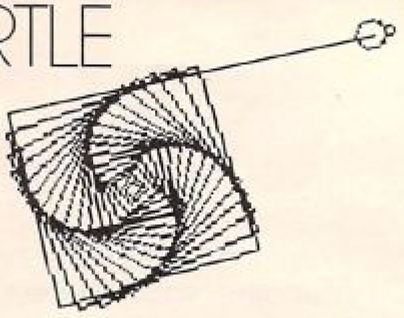
— ALL PRICES IN U.S. CURRENCY
— VISA OR MASTER CARD: SEND NUMBER, EXPIRY DATE, BANK NAME & CARD HOLDER'S NAME, INCLUDE PHONE NUMBER.
— CHECKS & MONEY ORDERS O.K. – CHECKS NEED THREE WEEKS TO CLEAR. (NO C.O.D.'S TO U.S. DUE TO POSTAL REGULATIONS)
— CANADIAN ORDERS ADD 25% FOR CDN FUNDS EXCHANGE.
— U.S. CUSTOMERS WILL HAVE TO PAY U.S. CUSTOMS 4.7% DUTY. DUTIES COLLECTED BY POST OFFICE. WE ASSUME NO LIABILITY FOR CUSTOMS CLEARANCE.

DEALER INQUIRIES REQUESTED

BYTE-RYTE
DEPT. CP
P.O. BOX 205, STATION CART.
MONTREAL, QUEBEC, CANADA H4K 2J5
(514) 335-1717

*Apple is a trademark of Apple Computer Inc.

*Ram-Tech is a trademark of 99506 CANADA INC.



David D. Thornburg, Associate Editor

Ed Emberley's Drawing Procedures

Part of the appeal of turtle graphics is that it allows complex pictures to be built from simple building blocks. This feature arises from the fact that each shape description or procedure describes the shape itself, independently of its starting point or orientation. For example, once a square is defined with the procedure:

```
TO SQUARE :SIZE
   REPEAT 4 [FORWARD :SIZE RIGHT 90]
END
```

the computer can use this procedure to create a square of any size at any starting location and orientation.

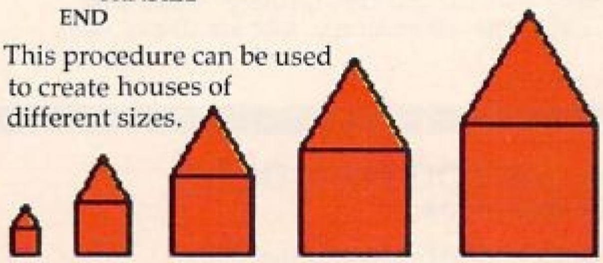
If the user has built up a set of useful geometric procedures, these can be combined to create more complex figures. If one also has a procedure for drawing triangles:

```
TO TRI :SIZE
   REPEAT 3 [FORWARD :SIZE RIGHT 120]
END
```

then a procedure for drawing a house can be created from a combination of a square and a triangle:

```
TO HOUSE :SIZE
   SQUARE :SIZE
   FORWARD :SIZE RIGHT 30
   TRI :SIZE
END
```

This procedure can be used to create houses of different sizes.



Many turtle graphics enthusiasts create libraries of basic figures from which quite interesting pictures can be created.

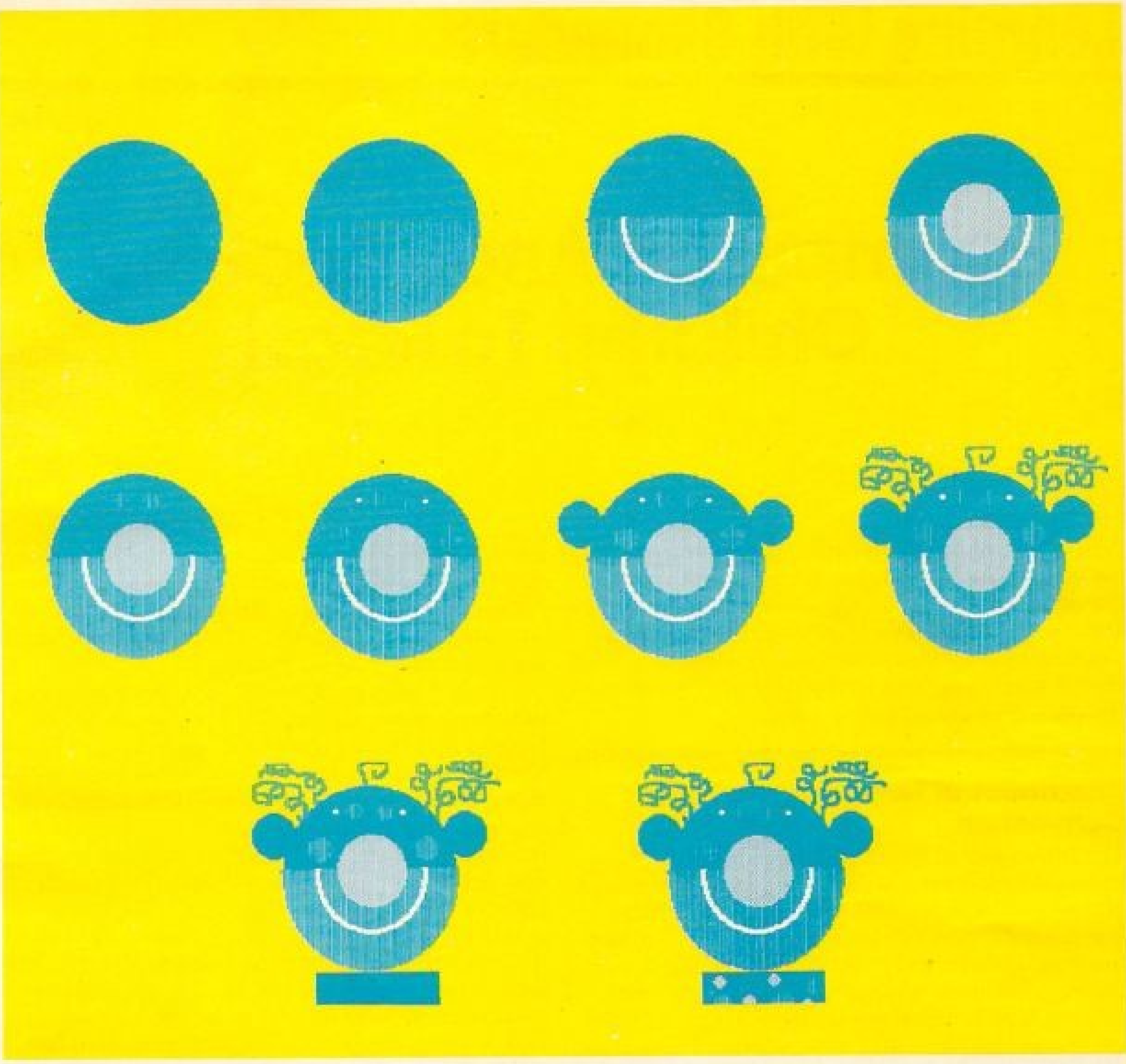
As an active proponent of turtle graphics and procedural problem-solving, I was delighted to find Ed Emberley's independent discoveries along these lines.

Ed Emberley has written several books on illustration for children. His books of particular interest to readers of this column would include: *Ed Emberley's Drawing Book of Animals*, *Ed Emberley's Drawing Book*, *Make a World*, *Ed Emberley's Big Orange Drawing Book*, and *Ed Emberley's Big Purple Drawing Book* (all published by Little, Brown and Co.).

Mr. Emberley's illustration technique is built on the idea that, just as words are created from an alphabet of letters, pictures can be created from an alphabet of shapes. He shows how to create myriad figures using circles, rectangles, arcs, lines, triangles, and other simple pieces. By building the figure piece by piece, the young artist is never overwhelmed by trying to deal with the whole figure at once. The following series of illustrations (courtesy of Mr. Emberley) shows how one can create a clown's head almost entirely from circles and circle parts.

If you were to create this figure using turtle graphics procedures, you would need only procedures for a circle, an arc, a rectangle, and the squiggles for the hair.

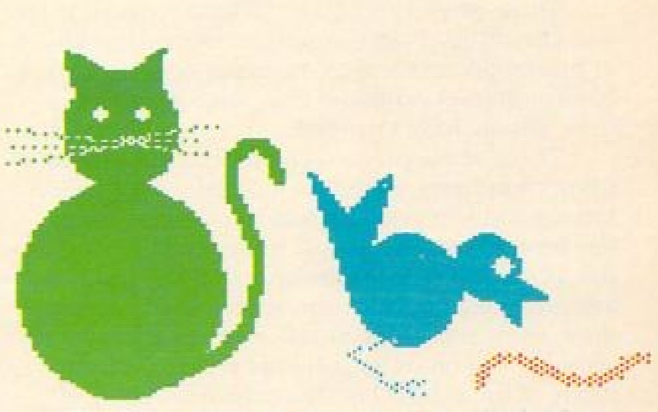
Ed Emberley does not normally use a computer to create his illustrations. The clown figures shown on the next page are a happy exception to that, as he created them on an Apple computer using the KoalaPad touch tablet with the Micro



Illustrator software. I am encouraging him to use Logo also to see how he likes it.

Just as Mr. Emberley's books can be a source of inspiration to those of us who build pictures using turtle graphics, they can also be wonderful tools for teaching procedural problem-solving – for teaching people how to solve larger problems by breaking them into bite-sized chunks. For this reason I encourage the use of his drawing books by teachers of computer programming. Not only are the children learning to solve problems with procedures, but they are also learning how to create charming illustrations at the same time.

I created the next figure myself to show that almost *anyone* can learn to make pictures in this manner.



For those of us who have been in the field a long time, the discovery of Mr. Emberley's excellent contributions is refreshing. Clearly, he is a Friend of the Turtle! ©

Apple Sounds – From Beeps To Music

Part 1

Blaine Mathieu

In this first of a two-part series, the author takes us from the simplest possible sound on the Apple to musical notes. Several useful demonstration programs are included.

Since I first acquired an Apple II+ about a year and a half ago, I have been fascinated by the strange noises I often hear. In this first of two articles I hope to save you all the trouble I went through in learning how to use APPLE sounds. Readers who already understand how to use CTRL-G and -16336 may want to skip the next section and go on to "Paddle Sounds."

Beeps And Clicks

Before you read this section, you should enter Program 1 on your computer and save it. Now run the program. If you entered it correctly, you should see SOUND #1 at the top, a line from the program, and a small menu.

Probably the first sound that you ever heard from your Apple's speaker was the bell sound. You can reproduce this in immediate mode by holding down the control key (labeled CTRL) and pressing the G key. In SOUND #1, you see that in line 30 a CHR$(7) is being printed – 7 is the numeric code for CTRL-G. If you are in Integer BASIC, you will have to use the format shown in line 35. In this line you'll see a PRINT with two quotes. Inside these quotes is a CTRL-G. The REM statement in line 37 shows how to type line 35. As you can see, control characters don't show up in a line listing or when you type them. An interesting side effect is that when you LIST your program, you will hear all the bell sounds in your program that are printed using the method in line 35.

In Program 1, the computer waits for you to hit a key. If you hit R, it will repeat any sound that might be produced by the above program lines. If you hit C, you will proceed to the next sound in Program 1. Any other key (except RESET) will cause no change.

Clicking -16336

Now hit C to go on to SOUND #2. In this program a simple FOR/NEXT loop is set up to beep the Apple's speaker ten times. Note the semicolon at the end of line 80; this prevents the screen from scrolling. If I hadn't used the semicolon, as each CTRL-G was printed the imaginary cursor would move down the screen until the screen started to scroll upward, which is, in most cases, undesirable.

Looking at SOUND #3, you will notice the number -16336, which is the memory address of the Apple's speaker. Every time this address is accessed, the Apple gives a little push on its speaker, creating a small click. PEEKing, as I have done in line 130, is just one simple way of accessing this address. If you missed the sound the first time, press R to hear it again.

SOUND #4 includes another simple loop that will PEEK the speaker's memory address 100 times. Instead of typing -16336 every time we wanted to use it, we assigned -16336 to the variable NO (for NOise). You may use any variable you wish, of course.

In SOUND #5 you'll notice line 250, which strings a lot of "clicks" together. This produces a longer noise than in SOUND #3 and a higher-pitched noise than in SOUND #4. As a rule, the closer your PEEKs, the higher-pitched your noise is going to be. In line 250 you will notice that we PEEKed -16336 a total of 15 times, a purely arbitrary number.

Finally, SOUND #6 demonstrates most of what we've learned about clicks. It uses a FOR/NEXT loop to cause line 320 to repeat 100 times. Line 320 has an assortment of minus and plus signs to show that it rarely makes a difference what you do to this location, as long as you access it.

Now on to something a little more exciting and complicated.

Paddle Sounds

Program 2 requires paddles or joystick. It is a simple BASIC program which reads a byte from the DATA statement and POKEs it into memory locations 768 ($300) to 786 ($312). The routine begins by CALLing 768. If you entered the program correctly, you should hear a fairly high-pitched whine, and as you move the paddles or joystick, this whine will change in pitch. You may leave the program by pressing RESET or CTRL-RESET, depending on your model.

Program 3 is the source code for this little machine language routine. Here is a quick explanation of the routine:

1. Put the paddle number in the X register.
2. Jump to the PREAD subroutine (see *Apple II Reference Manual*). PREAD acts as a delay, dependent on the paddle setting.
3. Tweak the speaker by accessing -16336 ($C030).
4. Repeat for next paddle.
5. Jump to beginning.

Since the pitch of the noise depends on how close together the tweaks are, the lower the paddle setting, the higher the pitch of the noise.

Making Music

Now we'll look at a program that lets you produce notes (and thus music) on your Apple. Of course, there are some limitations. For example, you won't be playing Beethoven's Fifth Symphony in five-part harmony with snare drum accompaniment. If you want that, many peripheral boards are available for the Apple which do amazing things. However, you can do quite a lot with the hardware already in your Apple.

Program 4 is a simple BASIC program that POKEs in a machine language subroutine, sets up a few parameters, and CALLs the subroutine. The program continues running until a key is pressed. Try running it. If you've never heard notes from your Apple, you will be quite pleased.

After the program has POKEd in the subroutine, it POKEs a random number (the pitch) into location 768 ($300) and POKEs a random number (the duration) into 769 ($301). The maximum value that can be POKEd into these locations is 255.

Program 5 is the source code for the "Note Producer" program that is POKEd into memory in Program 4. In essence, the program works much like "Paddle Sounds." The main difference is that instead of the paddles controlling the pitch of the sound, locations 768 and 769 control the pitch and duration of the tone. The source code contains comments that should help you understand what is happening.

As you can see by now, whenever you want a sound routine, you're going to have to access location -16336 ($C030). Try experimenting with Program 5 by POKEing in your own note values and hearing the results.

Next month, we'll look at a program called "Apple Music Writer," which will enable you to edit and play your own song. Until then, experiment with the programs here, and you're sure to come up with some surprising results.

Program 1: Sounds And Variations

```
5  REM  PROGRAM#1
10 I = 10: HOME
20  PRINT "SOUND #1": PRINT : LIST 30,3
    7
30  PRINT  CHR$ (7)
35  PRINT "": REM  CTRL-G
37  REM  PRINT"CTRL-G"
40  GOTO 10000
50 I = 50: HOME
60  PRINT "SOUND #2": PRINT : LIST 70,9
    0
70  FOR LOOP = 1 TO 10
80  PRINT  CHR$ (7);
90  NEXT
100  GOTO 10000
110 I = 110: HOME
120  PRINT "SOUND #3": PRINT : LIST 130

130 X =  PEEK ( - 16336)
140  GOTO 10000
150 I = 150: HOME
160  PRINT "SOUND #4": PRINT : LIST 170
    ,200
170 NO =  - 16336
180  FOR LOOP = 1 TO 100
190 X =  PEEK (NO)
200  NEXT
210  GOTO 10000
220 I = 220: HOME
230  PRINT "SOUND #5": PRINT : LIST 240
    ,260
240 NO =  - 16336
250 X =  PEEK (NO) +  PEEK (NO) +  PEEK
     (NO) +  PEEK (NO) +  PEEK (NO) +  PEEK
     (NO) +  PEEK (NO) +  PEEK (NO) +  PEEK
     (NO) +  PEEK (NO) +  PEEK (NO) +  PEEK
     (NO) +  PEEK (NO) +  PEEK (NO) +  PEEK
     (NO)
260  REM  FIFTEEN TIMES
270  GOTO 10000
280 I = 280: HOME
290  PRINT "SOUND #6": PRINT : LIST 300
    ,330
300 NO =  - 16336
310  FOR LOOP = 1 TO 100
320 X =  PEEK (NO) -  PEEK (NO) +  PEEK
     (NO) -  PEEK (NO) +  PEEK (NO) -  PEEK
     (NO) +  PEEK (NO)
330  NEXT
10000  POKE  - 16368,0: VTAB 20: HTAB 1
     : CALL  - 958: PRINT "'R' FOR REPE
     AT, 'C' TO CONTINUE ";: GET A$
10010  IF A$ <  > "R" AND A$ <  > "C" THEN
     10000
10020  IF A$ = "C" THEN 10100
10030  IF I = 10 THEN 30
10040  IF I = 50 THEN 70
```

```
10050  IF I = 110 THEN X =  PEEK ( - 16
    336): GOTO 130
10060  IF I = 150 THEN 170
10070  IF I = 220 THEN 240
10080  IF I = 280 THEN 300
10100  IF I = 10 THEN 50
10110  IF I = 50 THEN 110
10120  IF I = 110 THEN 150
10130  IF I = 150 THEN 220
10140  IF I = 220 THEN 280
10150  TEXT : HTAB 1: PRINT "END OF PRO
    GRAM#1"
```

Program 2: Paddle Sounds

```
10  REM  PROGRAM#2
20  FOR LOC = 768 TO 786: READ BYTE: POKE
     LOC,BYTE: NEXT LOC
30  DATA 162,0,32,30,251,141,48,192,162
     ,1,32,30,251,141,48,192,76,0,3
40  CALL 768
```

Program 3: Source Code For Paddle Sounds

```
           ORG   $300        ;768 DECIMAL
**********************************
*                                *
*PROGRAM#3 -   PADDLE     SOUNDS *
*                                *
**********************************
PDLZERO    EQU   $00
PDLONE     EQU   $01
PREAD      EQU   $FB1E
SPEAKER    EQU   $C030
START      LDX   #PDLZERO    ;SET UP FOR PADD
                             LE ZERO
           JSR   PREAD       ;GET DELAY FROM
                             PADDLE ZERO
           STA   SPEAKER     ;TWEAK SPEAKER
           LDX   #PDLONE     ;REPEAT P
                             ROCESS FOR PADDL
                             E ONE
           JSR   PREAD
           STA   SPEAKER
           JMP   START       ;START OVER
```

Program 4: Note Producer

```
5   REM  PROGRAM#4
10  FOR LOC = 770 TO 790: READ BYTE: POKE
     LOC,BYTE: NEXT
20  POKE 768, INT ( RND (1) * 255) + 1:
     POKE 769, INT ( RND (1) * 100) +
     1: CALL 770:X =  PEEK ( - 16384): IF
     X < 127 THEN  POKE  - 16368,0: GOTO
     20
30  DATA  173,48,192,136,208,5,206,1,3,
     240,9,202,208,245,174,0,3,76,2,3,96
40  POKE  - 16368,0
```

Program 5: Source Code For Note Producer

```
           ORG   $300        ;768 DECIMAL
**********************************
*                                *
*PROGRAM#5 -   NOTE.PRODUCER     *
*                                *
**********************************
TWEAK      EQU   $C030
PITCH      EQU   $300
DURATION   EQU   $301
           DS    2           ;MAKE SPACE FOR
                             PITCH AND DURATI
                             ON
START      LDA   TWEAK       ;TWEAK THE SPEAK
                             ER
BRANCH1    DEY
           BNE   BRANCH2
           DEC   DURATION    ;DURATION = DURAT
                             ION-1
           BEQ   RETURN      ;IF DURATION=0 T
                             HEN RETURN
BRANCH2    DEX
           BNE   BRANCH1
           LDX   PITCH
           JMP   START       ;CONTINUE TO SOUN
                             D NOTE
RETURN     RTS               ;GO BACK TO OPERAT
                             ING SYSTEM
```

©

• VIC 20 • COMMODOR 64 • ATARI • APPLE • SINCLAIR • VIC 20 • COMMODOR 64 • ATARI • APPLE • SINCLAIR •

THERE IS STRENGTH IN NUMBERS
JOIN
THE SOFTWARE CO-OP

NOW! For the cost of a single game cartridge you can join THE SOFTWARE CO-OP. Use the advantage of bulk-purchasing and pay **only $1 over whole-sale** for games, utilities and educational software. Rock-bottom prices on all equipment and supplies. Savings up to 40%. Guaranteed. Specializing in VIC 20, Commodor 64, Atari, Apple and Sinclair.

Write today for free details about our exciting new catalog and other sensational Co-op benefits including special swap system and free technical assistance.

THE SOFTWARE CO-OP
PO BOX 275 ELIZABETH, NJ 07207

COMSTAR AIR* SHIPPING WITHIN 2 DAYS
SEND $1.00 FOR COMPLETE LIST

VIC=20

16K RAM	$59
CARDBOARD (3 SLOT EXP)	33
BANANA PRINTER (80 COLUMN)	225
DISPLAY MANAGER (40/80 COL.) (WITH WORD PROC.)	89
RABBIT (EASTERN HOUSE) (VIC OR 64)	35
HES MODEM (WITH SOFTWARE) (VIC OR 64)	59
HES MON ASSEMBLER (C) (VIC OR 64)	29
DUST COVER (VIC, 64, 800, 400, 810, or 410)	7
QUICK BROWN FOX (C) (VIC OR 64)	54
SWORD OF FARGOAL (T) 21K	23

COM-64

WORDPRO 3 + (D)	69
VIDEOPAK 80 (80 COLUMN)	159
Z-80 VIDEOPAK (WITH CPM)	259
6502 PROF. DEV. SYSTEM (T)	23
PRACTICALC SPREAD SHEET (D)	42
LASER STRIKE (T)	22
ELEMENTARY 64 (BOOK)	14
TOTL LABEL (T) (VIC OR 64)	17
JUMP MAN (D,T)	29
ROBBERS OF LOST TOMB (D,T)	21
FORT APOCALYPSE (D,T)	26
JAWBREAKER (C)	26
FROGGER (D,T)	26
ANNIHILATOR (T)	18
TEMPLE OF APSHAI (D,T)	29

ATARI

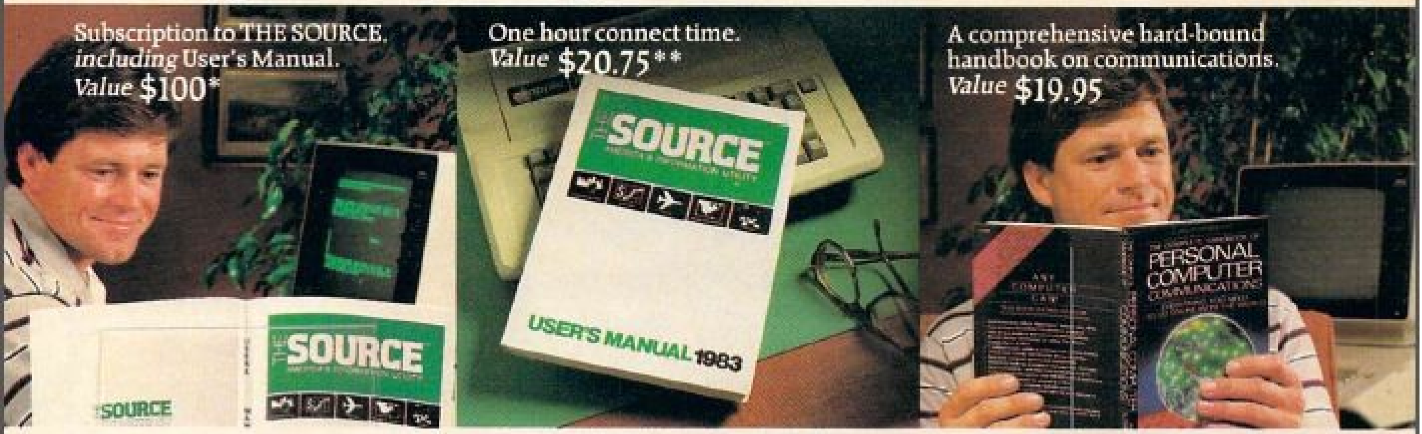
64K RAM (FOR 400)	109
48K RAM (FOR 400)	89
MONKEY WRENCH II	52
JUMP MAN JR. (C)	29
ALIEN GROUP VOICE BOX (D) 40K	119
TECHNICAL NOTES	25
NEWPORT PROSTICK	23
MICROBITS MODEM	159
MICROBITS PRINTER INTERFACE	83
DE RE ATARI (BOOK)	18
VAL FORTH (D) 24K	45
STAR LEAGUE BASEBALL (D,T) 24K	25
DONKEY KONG (C)	37
DIG DUG (C)	34
ALOG PAGEWRITER (D) 32K	34

C = CARTRIDGE D = DISK T = CASSETTE * MOST ITEMS

COMSTAR
P.O. BOX 1730 GOLETA, CA 93116
(805) 964-4660

ORDERS ONLY: 800-558-8803
or send check or money order. VISA, MC add 3%. Shipping—$2 for software (call for hardware). Calif add 6% tax. COD add $2.50.

Buy Micromodem IIe™ with Smartcom I™ software *right now,* and get this $140 value FREE!



Subscription to THE SOURCE, *including* User's Manual. *Value* $100*

One hour connect time. *Value* $20.75**

A comprehensive hard-bound handbook on communications. *Value* $19.95

Hayes

Your Apple® 's telephone.

Between Nov. 1–Jan. 15, you and your Apple can make the big break. From isolated desktop computing. To the exciting world of telecomputing. With Hayes Micromodem IIe and Smartcom I communications software. A complete telecomputing package for Apple II, III, IIe or Apple Plus computers. Let Micromodem IIe connect you, via telephone lines, to computers, terminals and information services all across America. Including THE SOURCE.℠ AMERICA'S INFORMATION UTILITY.℠

This offer takes you right to THE SOURCE! And you won't have to pay to join! The same day you purchase your Micromodem IIe with Smartcom I, call THE SOURCE on their toll-free number. And you're already well on your way to getting on-line!

It frees up your time. This offer also entitles you to $20.75 worth of connect time–at no charge. Use it as you please! Check the latest news and sports. Look up your flight schedule. View your stock portfolio. You name it. The time is yours! Sixty minutes worth, or more, depending on the day and hour.

Giving you lots of connections. THE SOURCE has assembled the most comprehensive programs available on any system. For fun and practical applications. From games to commodity news. Electronic mail to teleconferencing. Discount shopping to abstracts of articles from business magazines. And everything you need to know is right there in your comprehensive, illustrated User's Manual. It's included with your free membership to THE SOURCE during this special, limited time offer from Hayes.

Plus the last word on communications. Now here's a book that delivers what it promises! In Alfred Glossbrenner's best seller, *THE COMPLETE HANDBOOK OF PERSONAL COMPUTER COMMUNICATIONS*, you'll find a thorough, informative study of microcomputer communications. Worth every cent of the $19.95 this hard-bound book sells for. And yours absolutely free! *If you act now!* See your dealer soon. Buy Micromodem IIe with Smartcom I, and get the best telecomputing system for your Apple. *Plus* a $140 value FREE.

Hayes®

Hayes Microcomputer Products, Inc.
5923 Peachtree Industrial Blvd., Norcross, Georgia 30092. 404/449-8791.

Here's all you have to do:

1. When you purchase your Micromodem IIe with Smartcom I, *save* your sales receipt and Hayes registration card (packed inside the box).
2. Pick up the phone and call THE SOURCE, on their toll-free number:

1-800-336-3366.

Tell THE SOURCE representative that you are participating in the special Hayes promotion, give the serial number of your Micromodem IIe (on the modem), and your credit card number (VISA, Master Card, or American Express).† You will get your password to THE SOURCE, right on the spot!

3. Within a week you will receive an agreement from THE SOURCE, along with your New Member Kit. Sign the agreement, and return it within 10 days to THE SOURCE, along with your sales receipt for Micromodem IIe/Smartcom I, and Hayes registration card. Remember, send no money. Your membership is free!
4. That's all it takes! Look for your User's Manual and free communications book within two weeks of receipt of the agreement, sales receipt and registration card.

†THE SOURCE requires a major credit card for billing of hourly connect time to individual members. Corporate members may apply for direct billing.

*Suggested retail price. **60 minutes or more connect time, depending on the day and hour.

The Bear facts:

Now, you can get…

1 NEW software programs featuring Stickybear™ – the hottest-selling computer bear in America.

2 The most sophisticated color graphics ever for the Apple® Personal Computer.

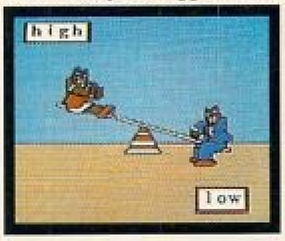
3 New arcade-quality games for families to play together.

4 New fun learning programs for 3- to 6-year-olds created by a world-famous children's author.

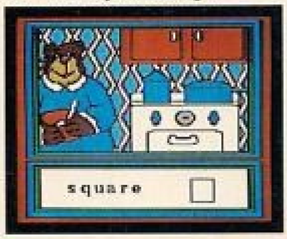
5 Disk, plus hard-cover book or game, poster and stickers in a sturdy, attractive vinyl binder – with each program.

Early Learning (Ages 3 to 6)

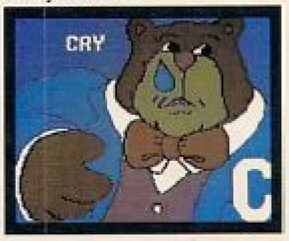
NEW Stickybear Opposites

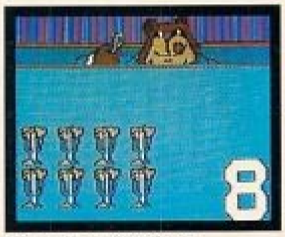


NEW Stickybear Shapes



Stickybear ABC

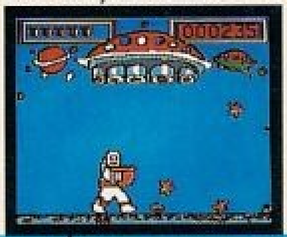




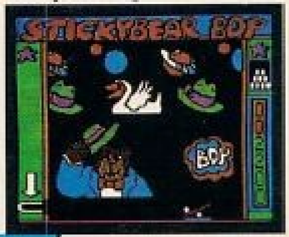
Stickybear Numbers

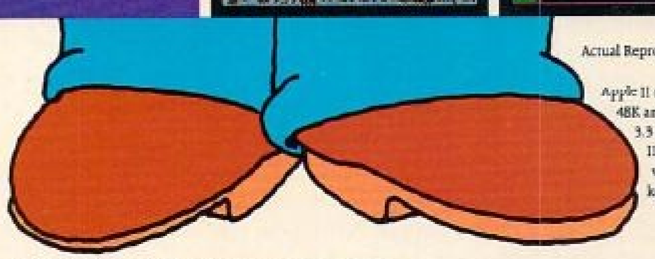
Games for the family

NEW Stickybear Basketbounce



Stickybear Bop





Actual Reproductions

Apple II or Apple II + with 48K and one disk drive, 3.3 DOS; and Apple IIe or Apple III. Use with paddles or keyboard.

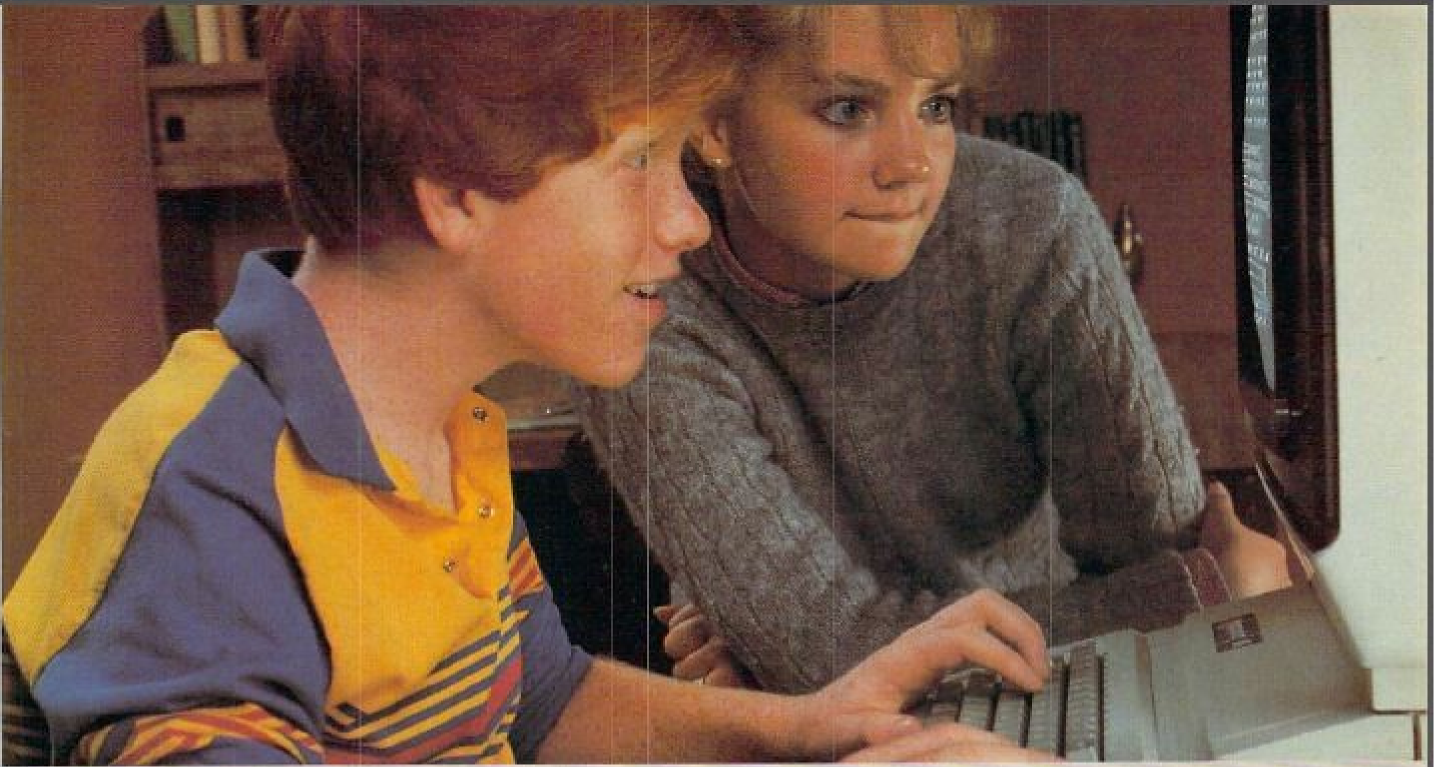
There's something new for everyone in your family! The playful animation and bubble-gum colors in the educational programs will captivate your youngest. While Stickybear Basketbounce and Stickybear Bop – games of skill and sharp wits – challenge even dedicated arcade games-players.

Look for Stickybear software – developed by Optimum Resource, Inc. for Weekly Reader Family Software – in finer computer stores everywhere. Or, call toll-free 1-800-852-5000, Dept. AB-33. Only $39.95 each.

Weekly Reader Family Software

A division of Xerox Education Publications
Middletown, CT 06457

A/M13-39



NOW, PLATO COURSEWARE MAKES ALGEBRA FRIENDLY.

Algebra can be a fascinating learning experience with the help of PLATO® educational courseware.

PLATO helps bring out the best in your child...and the best in your Apple II Plus or Apple IIe.
A new series of PLATO lessons can help your kids feel comfortable and confident about Algebra. Practice problems change at random to challenge them; examples of solutions guide them; performance reports encourage them. If a mistake is made, PLATO shows kids the correct answer or how to correct it. So before they know it, they're into Roots, Polynomials, Factoring, Equations and much more.

These Elementary Algebra lessons join the growing PLATO library of quality educational courseware for microcomputers. Other lessons include Math, Foreign Languages, Physics-Elementary Mechanics, Computer Concepts, Computer Literacy and Keyboarding.

See the PLATO line at selected retail outlets. All PLATO micro courseware is available for the Apple II Plus and Apple IIe. Some lessons are also available for the TI99/4A and Atari 800.
For a free PLATO catalog:
Call toll-free: 800-233-3784.
(In Calif., call 800-233-3785.)
Or write Control Data Publishing Co., P.O. Box 261127, San Diego, CA, 92126.

Warranty available free from Control Data Publishing Co., 4455 Eastgate Mall, San Diego, CA 92121.

PLATO
COMPUTER-BASED EDUCATION

CONTROL DATA
PUBLISHING

Androbot's Topo

Michael A. Tyborski

Robots are rapidly becoming part of our life. You cannot read a magazine or newspaper without hearing about them. Although robots were once laboratory curiosities, they are now within anyone's reach.

Yes, *you* can own a robot. Mechanical servant? Not yet. Entertaining companion, yes.

Androbot, Inc., of Sunnyvale, California, has recently released its Topo robot. It will provide hours of entertainment for any Apple owner. Although not a true robot, it demonstrates many of the important fundamentals of robotics.

Your Computer Controls It

Topo is a radio-controlled platform that looks like a robot. It includes a user's manual, transmitter, and plug-in control card for the computer. It also includes TopoBASIC on disk, which allows it to be used within a few minutes.

An Apple computer controls Topo; this simplifies programming and reduces the selling price. It also eliminates the need to learn a new operating system or programming language. Unfortunately, your computer does not receive sensor information, a limitation that makes it possible for Topo to run into walls or down the stairs.

The robot is made of high-impact plastic and is three feet tall. Its friendly appearance attracts small children like ice cream, an effect consistent with Androbot's belief that robots should be "friendly looking, inviting companions."

Topo has a head and arms. Unfortunately, they are not functional. The head is permanently attached to the body and does not turn, which makes the robot less lifelike. It has a decorative face grill and eyes. An emergency stop switch is mounted on top of the head which turns off the robot.

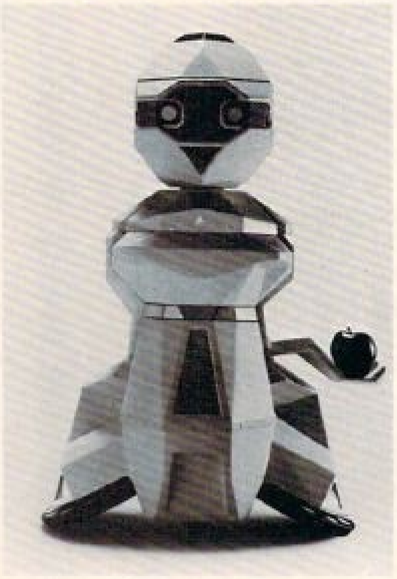
The arms, plastic flaps that can be extended as needed, are made from relatively thin plastic and cannot hold heavy loads. They attach to the body with plastic pins.

Two-Wheel Drive System

Topo has a unique drive system called Andromotion. Androbot claims that this provides "maximum stability and safety with optimum maneuverability and control." It also gives the robot an individual personality.

Just what is Andromotion? It is a two-wheel drive system that relies on angled wheels for stability. This design is patterned after the rocking chair. As a result, the robot remains stable because the effective roll center is above the center of gravity. The principle is clearer when the robot is viewed from the side. The side projection of the wheels looks like an ellipse, and the long sides resemble the rail of a rocking chair.

Because of Andromotion, Topo sways from front to back as it moves. This sway can become violent during a fast stop, making Topo look like a fishing bobber.


Androbot's Topo robot.

Androbot states that Topo has industrial-grade batteries and a fabricated steel superstructure, and claims that high-torque motors and cast aluminum gear boxes assure structural integrity. These features place the robot above the toy category.

The robot's back panel holds the power switches, indicator lights, and a charger jack. Yes, switches. For some reason, Androbot decided to use a separate ON and OFF switch, a design possibly based on a control circuit restriction. The red and green switches may also indicate STOP and GO to children.

The indicator lights show when Topo is on and what the battery status is. When a low voltage condition occurs, a red indicator light turns on. The wheel supports also contain indicator lights for showing direction.

You are responsible for plugging in the charger – a simple AC adapter. You must also prevent the robot from being turned on while charging. If it is, you may soon need a new charger. Finally, you must not leave the charger connected for more than 24 hours at a time.

Topo receives commands over a radio link. This link uses a 100-milliwatt, 4-channel AM transmitter that operates at 27.145 megahertz, and transmits the control card data. Although the antenna is short, a 90-foot range is possible. The transmitter has its own power switch to prevent interference when Topo is not being used.

The control card provides power and serial data for the transmitter. It plugs into slot five on the Apple computer. The unit has three integrated circuits and one regulator. This allows a 3-inch-square board to hold all the circuitry. An AMD 9513 chip generates the serial data for the transmitter.

The Documentation

The Topo manual is easy to read and understand. It comes in a small ring binder and includes dividers for future chapters. A plastic holder protects the program disk and warranty card. Interestingly, the manual was printed on a dot matrix printer, but this does not decrease its readability.

After an introduction to Androbot and Androbots, the user is shown how to unpack and check Topo. The first section also includes control card installation and battery charging instructions.

The important calibration procedure, which insures accurate movement and turning, is covered next. Finding calibration values for each surface Topo will move on will minimize errors from wheel slippage.

Finally, the last section describes TopoBASIC, and has material for the beginning and advanced programmer. This section includes a listing of the machine language and BASIC routines. It also provides a glossary of BASIC routines.

Topo In Motion

After charging the batteries, we began to use Topo under program control. This proved to be an interesting experience. Topo just did not like repeating its path. While drawing a square, for example, it turned about 15 degrees each repetition. This made the square rotate about its center.

Proper calibration improved its performance. In our case, the procedure took about ten minutes. It had to be repeated, however, for other surfaces.

The transmitter could control Topo throughout a house. It did have some annoying dead spots, however, which made Topo act erratically or stop.

Topo cannot detect obstacles. As a result, it often ran into people or furniture. This, in turn, changed its path or completely stopped it. Whenever this happened, it had to be stopped and moved to its starting point. The program was then restarted.

Spectator reactions varied. Adults and teenagers were either amused or skeptical. Many wondered what Topo could be used for. Young children, naturally, were a captive audience. They would try touching Topo whenever it stopped. Some even talked to it.

Having already seen Heathkit's Hero robot, many people missed voice and head movement, claiming that these features make robots interesting and lifelike. A few people also wanted the arms to move. Despite these objections, they all gave Topo a favorable rating.

Future Enhancements

Androbot will offer a number of accessories for Topo, including a voice module and Androwagon.

Sirius Is GIVING AWAY Over $35,000.00 In VIDEO GAMES!!!

You Can Win A Sirius Game OF YOUR CHOICE!

Correctly answer the questions below to qualify for the FREE GAME DRAWING. Enter all of the Grud Quizzes and qualify for all 20 drawings!

GRUD QUIZ #4: TYPE ATTACK

1. In which level does the word "REFER" appear?
2. What typing speed should genetic mutants and robots choose?
3. How much energy is lost for typing a wrong letter?
4. How do you PAUSE the game?

TO ENTER:

Just send a postcard or 3 x 5 piece of paper to Sirius containing the following information:
1. Your name and address
2. The Grud Quiz number and your four answers

RULES:

1. All entries must be handwritten and individually mailed.
2. All entries must be postmarked by midnight, February 29, 1984.
3. Only one prize per person per quiz is permitted.

1,000 WINNERS!

If ALL of your answers are correct you will qualify for that Grud Quiz drawing. 50 winners per quiz will be randomly chosen and notified by mail on or before March 31, 1984. Each winner will receive a complete Sirius catalogue listing from which they can select the game or games of their choice.

LOOK FOR MORE GRUD QUIZZES in the Oct., Nov., and Dec. issues of your favorite computer magazines or at participating software retailers.

ALL CONTESTANTS WILL RECEIVE A FREE FULL COLOR SIRIUS COMIC CATALOGUE

Sirius™

Sirius Software, Inc.
10364 Rockingham Drive
Sacramento, CA 95827

(VOID WHERE PROHIBITED BY LAW)

The voice module will use a tape recorder for high-quality, low-cost speech. It will be controlled by a spare transmitter channel and should help attract spectators and hold their attention.

The Androwagon compensates for the cosmetic arms. It allows Topo to carry beverages and other heavy objects. When combined with speech, this accessory could turn Topo into a party host.

Programmers should look forward to working with TopoLogo and TopoForth. These languages simplify program development and allow commands such as GO KITCHEN. They also draw the path Topo is currently following. A TEACH mode saves time by allowing command sequences to be created and saved on disk for future recall.

TopoLogo consists of extensions to Terrapin and Krell Logos for the Apple II computer, and includes calibration and demonstration programs. This package provides the most powerful way to control Topo.

Finally, interface boards will soon be available for Atari, Commodore, and Radio Shack computers. This will undoubtedly make Topo more visible. Despite its limitations, Topo provides an excellent introduction to robotics.

Topo
Androbot, Inc.
101 E. Daggett Drive
San Jose, CA 95134
$795
$495 without sound ©

Paper Porter

Betsy and Stefan Burr

There's something particularly attractive about a simple, inexpensive device that claims to do the work of complicated hardware. That's what intrigued us about a piece of plastic called the Paper Porter, which is designed to give friction-feed capability to a tractor-feed printer such as the Epson MX-80. Since friction feed can add as much as $100 to the cost of a printer, this alternative, at less than $5, is worth considering.

The Paper Porter is a 9½-by-17-inch sheet of clear plastic with holes punched in the side so that it can be driven by tractor pins. Near the top is a pocket formed by another sheet, which can hold by friction an ordinary piece of paper, such as letterhead. Once the top of the paper is inserted into the pocket, the Paper Porter is easily loaded and run through the printer. The procedure is repeated with each page in a multipage document. With practice, we found that the whole operation takes only a few seconds – quite comparable to the time needed to run each separate sheet through a printer with friction feed.

Print On Letterheads Or Ditto Masters

Although printing on letterhead stationery may be its primary use, the Paper Porter can come in handy in other ways. For example, we use it to make ditto masters.

A minor difficulty arises in trying to print close to the top of a page. The plastic pocket overlaps the top of the paper by one inch, making it impossible to print above that point. We solve this problem by putting two small loops of masking tape, sticky side out, in the pocket. This holds the page so that printing can start within half an inch or so of the top. After you put the tape in place, you may need to reduce the stickiness a bit. Double-stick tape works, too, but it's a trifle harder to adjust the stickiness. Once the tape is properly placed and adjusted, the fix lasts for months.

On letterhead stationery, of course, there is no need to come near the top of the page, so it can be useful to have one Paper Porter with the tape and one without it. We've ended up acquiring two of each type, so we can be slipping one page into a Paper Porter while the other is printing – a timesaver on multipage jobs.

The 17-inch length is just enough to prevent the out-of-paper switch on the Epson MX-80 from terminating printing before the bottom of an 11-inch page. An earlier version of the Paper Porter was too short, making it impossible to print to the bottom of a page unless the switch had been defeated. With paper longer than 11 inches, and perhaps with some other printers, the alarm may still be activated. And, just as with some platen feed arrangements, the alarm may come on when pages are being changed. For these reasons, it may be desirable to defeat the switch, which is not usually difficult. On the MX-80 it can be done by taping a small piece of paper over the switch.

Business Envelopes Not Compatible

The Paper Porter does have one significant drawback: it can't print on a standard business envelope. Any paper that is even slightly wider than 8½ inches will interfere with the tractor pins.

There is at least one trick which is actually easier with the Paper Porter than with a typical friction (platen) feed printer – printing two or more columns in perfect alignment. The standard procedure is to print one column, then back up the paper and print the second column. With friction feed, the alignment is tricky, but with tractor feed, the pins guarantee that the backed-up page can be perfectly aligned with no trouble.

The Paper Porter
5718 Ponderosa Drive
Stevensville, MI 49127
(616) 429-6461
$4.50 ($3.50 in lots of five) ©

Apple Sounds — From Beeps To Music

Part 2

Blaine Mathieu

In the conclusion of this two-part series, the author combines the ideas and programs from Part 1 and presents the "Apple Music Writer." An effective tool for composing or reproducing songs, this utility is also easy to use because of its great variety of commands. There's a thorough discussion of how to use each command.

"Apple Music Writer" is a program which will allow any Apple owner to easily reproduce his or her favorite songs. When you run the program, the first thing you'll notice is the title, and then you'll hear part of a tune that you may recognize. After the tune stops, you will be prompted by the word COMMAND? and a flashing cursor. At the top of the screen you should see a list of the possible commands and corresponding letters. On the right side of your screen you should see a list of note names with corresponding values.

It's important that you understand and know how to use the commands, so let's review them in some detail, in the order that they appear on the screen. These commands are usable only from the COMMAND? mode; you must also RETURN after each command. You may want to experiment with them as we go along.

The Commands

A = ADDNOTE. This command will let you begin your music file (song) and keep adding to it. Every time you press A (and RETURN) you will be prompted to enter the note, a comma, and the duration. For example:

```
NOTE#1
NOTE,DURATION 128,200
```

The maximum note value is 255 (actually 0 = 256). The same is true for the duration value. After you've entered your values, you will hear what the new note will sound like in the song.

E = EDIT. If you've made a mistake, you can fix it by typing E (and, as always, RETURN). You will then be asked the number of the note you want to edit. If the note you want to edit is not part of the music file, you will be reprompted for the note number. If you entered a valid note number, you will be given the old values for that note and prompted for new values. The same rules apply for entering data as in ADDNOTE. Let's say you want to edit note number one and replace the old values with new ones of 64 and 200:

```
COMMAND? E
EDIT NOTE#1
NOTE#1 OLD: NOTE=128 DUR=200
NOTE,DURATION: 64,200
```

P = PLAY. Typing P will put you into Play mode. This will play your song and print it to the screen at the same time. Because it is both listing and playing your music file, the playing will not be at the same speed as in your program. It will be slower and more pronounced. After entering P you will be prompted for the starting and ending note to Play/list. If you just press RETURN instead of entering values, the whole song will be played (defaults will be set; D is the default).

S = SAVE. This command will SAVE your music file to disk. First you will be prompted for a filename, which will be the name used when the file is SAVEd. Then you'll be prompted for the number of the first and last note of your file that you want saved to disk. The next question is FOR FUTURE ADDITION? A little explanation is in order here. There are two types of files which can be produced with this command. If you answer Y to the above question, a file will be created that can be reloaded into Apple Music Writer at any time. You should use this option if you feel you may want to add more notes or edit your song at a later date. If you enter N, a file will be created that you can easily turn into a BASIC program that will play your song when run.

If you answer the FUTURE ADDITION? question with an N, you will be asked for the starting line number of your soon-to-be-created BASIC music program. Then you will be asked if you want a FULL LOADER PROGRAM? If you answer Y, the BASIC program created will include the necessary information so that when your new program is RUN, the machine language "Note Producer" (see Part 1) routine will be POKEd in. If you answer N, the routine will not be included. You would answer N if the program you wanted to add the music to already included some sort of "Note" routine (the routine found in Program 5 of Part 1 of "Apple Sounds – From Beeps To Music").

Finally, you will be prompted to check for errors. If everything is all right, enter Y and the file will be SAVEd. If you enter N, you have to repeat the entire SAVE process. Here is an example of what the average SAVE command might encompass:

```
COMMAND? S
(Screen is cleared)
FILENAME? SONG.1
STARTING NOTE NUMBER: 2
ENDING NOTE NUMBER: 10
FOR FUTURE ADDITION? N
STARTING LINENUMBER: 100
FULL LOADER PROGRAM? Y
IS EVERYTHING OK? Y
```

Your music file would now be SAVEd under the filename SONG.1. The file would consist of notes two through ten, and the generated program would start at line 100. The generated program would include the machine language "Note" routine.

L = LOAD. If you answered Y to the FOR FUTURE ADDITION? question back in the SAVE command, you can LOAD an old music file back into the computer. The catch is that you will lose any data that you entered into the computer beforehand. If you don't want to lose your data, then answer N to the question about losing your data. Just enter the appropriate filename, and you can manipulate or add to your data once again.

N = NORPLAY. As mentioned earlier, when you P (Play/list) your song, it will play at a slower speed because it has to list the note values at the same time. To alleviate that problem, you can use the NORmal PLAY command. This will play your song in the same tempo as it will normally be played by your generated program. Just enter the proper values (or defaults will be used) and listen.

D = DELETE. Upon entering D from the COMMAND? mode, you will be asked which note or notes you want to delete. If you hit RETURN after the first question without typing anything else, the default will be used and the last note in the music file will be deleted. If you enter a value for the first question, you will be asked the number of the last note up to which you want to delete. The appropriate notes will then be deleted, and you're back to the COMMAND? mode.

I = INSERT. This command is the exact opposite of the Delete command. Simply answer the few setup questions and enter the data. Note: You cannot leave the Insert mode until you have entered all the data you specified you were going to enter.

R = RESTART. This command lets you start over with a clean slate beginning with note number one.

C = CATALOG will return a fairly standard DOS catalog.

Q = QUIT. Use this command to exit the program cleanly. You will lose all your data that hasn't been SAVEd to disk. If you quit by accident, a GOTO 200 will usually let you reenter the program with no data lost.

. = DOS. What this means is that typing a period followed by any normal DOS command will execute that command. A common use for this might be:

```
COMMAND? .DELETE FILENAME
```

Caution: Certain DOS commands will cause the Apple Music Writer to cease functioning, thus causing a loss of data. Take care.

H = HARD. If you have a printer connected to your Apple, you can get a simple hard copy of your music file by entering H from the COMMAND? mode. Note: You may have to edit lines 1210 and 1220 to accommodate different printers.

Hints For Easier Use

Saving. One good idea is to save two copies of your music file to the disk. One copy should be done in the FUTURE ADDITION? mode so you can edit or add to it at a later date. If you wish, the other copy can be done in the *create program*, or FUTURE ADDITION? N mode. Always remember to use a different filename.

Tempo. When you enter your durations, remember that if your quarter note has a value of 50, your half note will have a value of 100 and so on. You should set a plan of what duration you want a certain type of note to be and work from there. Rests are done with a note value of one.

Limits. The number of notes you can have in one song is limited. For our purposes the number is 500, but by changing the value of L in line 120, this limit can be raised.

Notes. The note listings on the side of the screen are especially helpful if you are transposing sheet music to disk. The numbers listed are for the middle octave. For the higher octave, divide the number by two; and for the lower octave, multiply the number by two. For example, the note F could be represented by the numbers 36, 72, and 144. You can also make a separate list of all the notes and their numbers. Remember, F# is the same as G-flat and so on. Also, once again, the number zero is equivalent to the number 256.

Exec. In order to use a program that you made in the FUTURE ADDITION? N mode, you must EXEC it. EXEC is a DOS command that prints a sequential text file to the screen as if it were typed from the keyboard. In this way, you can EXEC your file and RUN it as a BASIC program. Later on, you can SAVE it. Another feature is that you can LOAD an old BASIC program (game or whatever) and EXEC your sound routine into it. For this to work properly, however, you must have specified the starting line number during the save of your music file such that the line numbers of the music routine do not conflict with those of the program to which the routine is being added.

Insert. If you have a large amount of repetitive data to type in, one trick is to enter the last note of that data, then Insert the rest. This saves you from repeatedly typing A from COMMAND? mode. (This is useful only if you know beforehand exactly what data you want to enter.)

Keys. There are a number of key codes that you can use with Apple Music Writer. If at any time the screen is getting too cluttered, an ESC-SHIFT-P should do the trick. You can stop a Catalog or a Play/list at any moment with a CTRL-S, and restart it with the touch of any key. Finally, in this program, CTRL-C RETURN can be a useful but sometimes dangerous command. I would recommend using CTRL-C only if you are caught in a never-ending loop or as a last resort. If for any reason you find yourself out of Apple Music Writer, you can usually reenter the program, without losing any data, by typing GOTO 200.

Experiment. No matter how long or well written a manual, nothing can take the place of hands-on experience with a program. Before you try any big projects, be sure you know what's going to happen at all times no matter what you enter during Apple Music Writer. Overall the program is very forgiving. One last thing – the best songs on the Apple seem to be songs with few or no rests. Try using longer notes instead of rests.

If you'd rather not type in the program, send $3, a stamped, self-addressed disk mailer, and an initialized blank disk (Apple DOS 3.3 compatible) to:

Blaine Mathieu
Box 2572
Peace River, Alberta
Canada, T0H 2X0

Apple Music Writer

```
10  REM    APPLE MUSIC WRITER
20  REM  INITIALIZATION
30  TEXT : HOME : VTAB 1: PRINT "A=ADDN
     OTE E=EDIT    P=PLAY    S=SAVE
     L=LOAD    N=NORPLAY D=DELETE  I=IN
     SERT  R=RESTART C=CATALOG Q=QUIT .
     =DOS H=HARD": PRINT "-------------
    ---------------------------": POKE
    34,5
40  VTAB 6: HTAB 34: PRINT "G =64": PRINT
     TAB( 34)"F#=68": PRINT  TAB( 34)"
    F =72": PRINT  TAB( 34)"E =76": PRINT
     TAB( 34)"D#=81": PRINT  TAB( 34)"
    D =86": PRINT  TAB( 34)"C#=91"
50  PRINT  TAB( 34)"C =96": PRINT  TAB(
    34)"B =1Ø2": PRINT  TAB( 34)"A#=1Ø
    8": PRINT  TAB( 34)"A =115": PRINT
     TAB( 34)"G#=121": PRINT  TAB( 34)
    "G =128": PRINT  TAB( 34)"/2 FOR":
     PRINT  TAB( 34)"HIGHER": PRINT  TAB(
    34)"*2 FOR": PRINT  TAB( 34)"LOWER
    ": POKE 33,32
6Ø  FOR LOC = 77Ø TO 79Ø: READ BYTE: POKE
    LOC,BYTE: NEXT
7Ø  DATA  173,48,192,136,2Ø8,5,2Ø6,1,3,
    24Ø,9,2Ø2,2Ø8,245,174,Ø,3,76,2,3,96
8Ø  HOME : INVERSE : VTAB 1Ø: HTAB 9: PRINT
    "APPLE MUSIC WRITER"
9Ø  FOR R = 1 TO 26: READ P,D: POKE 768
    ,P: POKE 769,D: CALL 77Ø: NEXT R
1ØØ  DATA 172,75,162,75,152,75,144,75,1
    Ø8,1ØØ,1,3Ø,144,75,1Ø8,1ØØ,1,3Ø,14
    4,75,1Ø8,255,1,1Ø,1Ø8,75,96,75,91,
    75,86,75,1Ø8,75,96,75,86,1ØØ
11Ø  DATA 1,1Ø,115,75,96,1ØØ,1,1Ø,1Ø8,1
    5Ø,144,15Ø,216,2ØØ,
12Ø  HOME :L = 5ØØ: DIM N(L),D(L),N$(L)
    ,D$(L),NN(L),ND(L)
13Ø  REM  MAIN ROUTINES START
14Ø  VTAB 5: GOTO 19Ø
15Ø I = I + 1
16Ø  PRINT : INVERSE : PRINT "NOTE#"I:
    NORMAL
165  INPUT "NOTE,DURATION ";N$(I),D$(I)
    : IF N$ =  CHR$ (8) OR D$ =  CHR$
    (8) THEN N$(I) = N$(I - 1):D$(I) =
    D$(I - 1)
17Ø N(I) =  VAL (N$(I)):D(I) =  VAL (D$
    (I)): IF N(I) > 255 OR N(I) < Ø OR
    D(I) > 255 OR D(I) < Ø THEN 16Ø
18Ø  POKE 768,N(I): POKE 769,D(I): CALL
    77Ø
19Ø  ONERR  GOTO 37Ø
2ØØ  PRINT : INPUT "COMMAND? ";A$
21Ø  IF A$ = "A" AND I = L THEN  PRINT
    "YOU ARE AT YOUR LIMIT!!!": GOTO 2
    ØØ
22Ø  IF A$ = "A" THEN 15Ø
23Ø  IF I <  = Ø AND (A$ = "E" OR A$ =
    "P" OR A$ = "H" OR A$ = "N" OR A$ =
    "I" OR A$ = "S") THEN  PRINT "SORR
    Y, NO NOTES":I = Ø: GOTO 19Ø
24Ø  IF A$ = "Q" THEN 45Ø
25Ø  IF A$ = "E" THEN 47Ø
26Ø  IF A$ = "P" THEN 39Ø
27Ø  IF A$ = "S" THEN 53Ø
28Ø  IF A$ = "D" THEN 141Ø
29Ø  IF A$ = "L" THEN 99Ø
3ØØ  IF A$ = "R" THEN I = Ø
31Ø  IF A$ = "C" THEN  PRINT  CHR$ (4)"
    CATALOG"
32Ø  IF  LEFT$ (A$,1) = "." THEN 112Ø
33Ø  IF A$ = "H" THEN 116Ø
34Ø  IF A$ = "N" THEN 125Ø
35Ø  IF A$ = "I" THEN 131Ø
36Ø  GOTO 19Ø
37Ø  PRINT "ERROR#" PEEK (222): GOTO 19Ø
38Ø  REM  PLAY ROUTINE
```

```
390  PRINT : INPUT "STARTING NOTE (D=1)
     : ";SN$:SN =  VAL (SN$): IF SN$ =
     "" THEN SN = 1
400  PRINT : INPUT "ENDING NOTE (D=LAST
     ): ";EN$:EN =  VAL (EN$): IF EN$ =
     "" THEN EN = I
410  IF SN < 1 OR SN > I OR EN < 1 OR E
     N > I THEN 390
420  PRINT : INVERSE : PRINT "START OF
     SONG": PRINT : NORMAL : FOR X = SN
      TO EN: POKE 768,N(X): POKE 769,D(
     X): PRINT "NOTE#";X;: HTAB 10: PRINT
     "NOTE=";N(X);: HTAB 19: PRINT "DUR
     ATION=";D(X): CALL 770: NEXT X
430  INVERSE : PRINT : PRINT "END OF SO
     NG": NORMAL
440  GOTO 190
450  TEXT : HOME : PRINT "GOODBYE": END

460  REM  EDIT ROUTINE
470  INPUT "EDIT NOTE# ";NN: IF NN > I OR
     NN < 1 THEN 470
480  PRINT : INVERSE : PRINT "NOTE#"NN;
     : NORMAL : PRINT " OLD: NOTE="N(NN
     );" DUR="D(NN)"
490  INPUT "NOTE,DURATION: ";N$(NN),D$(
     NN):N(NN) =  VAL (N$(NN)):D(NN) =
      VAL (D$(NN)): IF N(NN) > 255 OR N
     (NN) < 0 OR D(NN) > 255 OR D(NN) <
     0 THEN 480
500  POKE 768,N(NN): POKE 769,D(NN): CALL
     770
510  GOTO 190
520  REM  SAVE ROUTINE
530  ONERR  GOTO 860
540  HOME : INPUT "FILENAME? ";FI$: IF
     FI$ = "" THEN 540
550  PRINT : INPUT "STARTING NOTE NUMBE
     R: ";SN: IF SN < 1 OR SN > I THEN
     550
560  PRINT : INPUT "ENDING NOTE NUMBER:
      ";EN: IF EN > I OR EN < 1 THEN 56
     0
570  PRINT : INPUT "FOR FUTURE ADDITION
     ? ";A$: IF A$ <  > "N" AND A$ <  >
     "Y" THEN 570
580  IF A$ = "Y" THEN  POKE 216,0:F2 =
     1: GOTO 640
590 F2 = 0
600  PRINT : INPUT "STARTING LINENUMBER
     : ";SL: IF SL > 63900 OR SL < 0 THEN
     600
610  PRINT : INPUT "FULL LOADER PROGRAM
     ? ";A$:A$ =  LEFT$ (A$,1): IF A$ <
      > "Y" AND A$ <  > "N" THEN 610
620  IF A$ = "Y" THEN FL = 1
630  IF A$ = "N" THEN FL = 0
640  PRINT : INPUT "IS EVERYTHING OK? "
     ;A$: IF  LEFT$ (A$,1) = "Y" AND F2
      = 1 THEN 880
650  IF  LEFT$ (A$,1) = "Y" AND F2 <  >
     1 THEN 670
660  GOTO 190
670 D$ =  CHR$ (4): PRINT D$"OPEN"FI$
680  PRINT D$"DELETE"FI$
690  PRINT D$"OPEN"FI$
700  PRINT D$"WRITE"FI$
710  IF FL <  > 1 THEN  GOTO 740
720  PRINT SL;"FORLOC=770TO790:READBYTE
     :POKELOC,BYTE:NEXT":SL = SL + 2
730  PRINT SL;"DATA173,48,192,136,208,5
     ,206,1,3,240,9,202,208,245,174,0,3
     ,76,2,3,96":SL = SL + 2
740  PRINT SL;"FORR=1TO";EN - SN + 1;":
     READP,D:POKE768,P:POKE769,D:CALL77
     0:NEXTR":SL = SL + 2
750  FOR Z = SN TO EN
760 N = N + 1: IF N = 20 THEN N = 1
770  IF N <  > 1 THEN 810
780  PRINT
790  PRINT SL;"DATA";
800 SL = SL + 2
810  PRINT N(Z);",";D(Z);: IF N <  > 19
      THEN  PRINT ",";
820  NEXT Z
830  PRINT
840  PRINT D$"CLOSE"
850  GOTO 190
860  PRINT : PRINT  CHR$ (7);"ERROR#"; PEEK
     (222): PRINT D$"CLOSE": GOTO 190
870  REM  2ND SAVE ROUTINE
880  ONERR  GOTO 980
890 D$ =  CHR$ (4): PRINT D$"OPEN"FI$
900  PRINT D$"DELETE"FI$
910  PRINT D$"OPEN"FI$
920  PRINT D$"WRITE"FI$
930  FOR S = SN TO EN
940  PRINT N(S): PRINT D(S)
950  NEXT S
960  PRINT D$"CLOSE"
970  GOTO 190
980  REM  LOAD ROUTINE
990  ONERR  GOTO 1090
1000  INPUT "YOU WILL LOSE YOUR DATA, O
     K? ";OK$:OK$ =  LEFT$ (OK$,1): IF
     OK$ <  > "Y" AND OK$ <  > "N" THEN
     1000
1010  IF OK$ = "N" THEN  POKE 216,0: GOTO
     190
1020  PRINT : INPUT "FILENAME: ";FI$: IF
     FI$ = "" THEN 1020
1030 D$ =  CHR$ (4): PRINT D$"VERIFY"FI
     $: PRINT D$"OPEN"FI$
1040  PRINT D$"READ"FI$
1050  FOR Z = 1 TO L
1060  INPUT N(Z): INPUT D(Z)
1070  IF N(Z) <  = 255 AND D(Z) <  = 25
     5 THEN  NEXT Z: POKE 216,0: PRINT
     D$"CLOSE":I = Z - 1: GOTO 190
1080  PRINT : PRINT "INCOMPATIBLE FILE!
     !!": PRINT D$"CLOSE": POKE 216,0: GOTO
     190
1090  PRINT D$"CLOSE": IF  PEEK (222) =
     5 THEN  POKE 216,0:I = Z - 1: GOTO
     190
1100  PRINT : PRINT "ERROR#"; PEEK (222
     ): PRINT D$"CLOSE": GOTO 190
1110  REM  HANDLE DOS COMMANDS
1120  ONERR  GOTO 1140
1130 DC$ =  RIGHT$ (A$, LEN (A$) - 1): PRINT
      CHR$ (4);DC$: POKE 216,0: GOTO 19
     0
1140  PRINT "ERROR#" PEEK (222): PRINT
      CHR$ (4)"CLOSE": POKE 216,0: GOTO
     190
1150  REM  PRINTER ROUTINE
1160  PRINT : INPUT "PRINTER READY? ";A
     $: IF A$ <  > "Y" AND A$ <  > "N" THEN
     1160
1170  IF A$ = "N" THEN 200
1180  PRINT : INPUT "STARTING NOTE TO B
     E PRINTED --  DEFAULT=1: ";ST$: IF
     ST$ = "" THEN ST$ = "1"
1190  PRINT : INPUT "ENDING NOTE TO BE
     PRINTED --    DEFAULT=ALL: ";EN$: IF
     EN$ = "" THEN EN$ =  STR$ (I)
1200 ST =  VAL (ST$):EN =  VAL (EN$): IF
     ST < 1 OR ST > I OR EN < 1 OR EN >
     I OR EN < ST THEN 1180
1210  PRINT : INPUT "NAME OF SONG: ";FI
     $: IF FI$ = "" THEN 1210
1220  PR# 1: PRINT : PRINT FI$: PRINT :
      FOR X = ST TO EN: PRINT "NOTE#";X
     ;: HTAB 10: PRINT "NOTE=";N(X);: HTAB
     19: PRINT "DURATION=";D(X): NEXT X

1230  PRINT : PRINT "END OF SONG": PR#
     0: GOTO 190
1240  REM  NORMAL PLAY ROUTINE
1250  PRINT : INPUT "STARTING NOTE (D=1
     ): ";SN$:SN =  VAL (SN$): IF SN$ =
     "" THEN SN = 1
1260  PRINT : INPUT "ENDING NOTE (D=LAS
     T): ";EN$:EN =  VAL (EN$): IF EN$ =
     "" THEN EN = I
1270  IF SN < 1 OR SN > I OR EN < 1 OR
     EN > I THEN 1250
1280  FOR Z = SN TO EN: POKE 768,N(Z): POKE
     769,D(Z): CALL 770: NEXT Z
1290  GOTO 190
1300  REM  INSERT ROUTINE
1310  POKE 216,0: PRINT : INPUT "INSERT
      BEFORE WHAT NOTE? ";IB: IF IB < 1
      OR IB > I THEN 1310
1320  PRINT : INPUT "HOW MANY NOTES TO
     INSERT? ";HM: IF HM > L - I OR HM <
     1 THEN 1320
1330  FOR Z = IB TO IB + HM - 1
1340  PRINT : INVERSE : PRINT "NOTE#"Z:
      NORMAL : INPUT "NOTE,DURATION: ";
     NN(Z),ND(Z): IF NN(Z) < 0 OR NN(Z)
      > 255 OR ND(Z) < 0 OR ND(Z) > 255
      THEN 1340
1350  POKE 768,NN(Z): POKE 769,ND(Z): CALL
     770
1360  NEXT Z
1370  FOR Z = I TO IB STEP  - 1:N(Z + H
     M) = N(Z):D(Z + HM) = D(Z): NEXT Z

1380  FOR Z = IB TO IB + HM - 1:N(Z) =
     NN(Z):D(Z) = ND(Z): NEXT Z
1390 I = I + HM
1400  GOTO 190
1410  REM  DELETE ROUTINE
1420  PRINT : INPUT "DELETE FROM NOTE (
     D=LAST): ";DF$: IF DF$ = "" THEN I
      = I - 1: IF I =  - 1 THEN I = 0: GOTO
     190
1430  IF DF$ = "" THEN 190
1440  PRINT : INPUT "TO NOTE: ";DT$:DF =
      VAL (DF$):DT =  VAL (DT$): IF DT <
     1 OR DT > I OR DF < 1 OR DF > I OR
     DF > DT THEN 1420
1450  FOR Z = DT + 1 TO I:N(Z - (DT - D
     F + 1)) = N(Z):D(Z - (DT - DF + 1)
     ) = D(Z): NEXT Z
1460 I = I - (DT - DF + 1): GOTO 190
```
©

EPSON*, NEC*, PROWRITER*, GEMINI*, OKIDATA 92*

NEW! Atari*

PRINTWIZ

The only self-booting grafix handler for dumps in **horizontal** format — all mach. lang. — **Lister** incl. — all modes — mixed modes — change aspect ratios, etc. while running other programs — assem ed — basic or no cartridge — demos, utilities, fonts, included — dump, create forms, stationery, calendars, requires interface. ONLY **$26.95**

diskwiz

Map, search, repair, duplicate, fix link pointers, alter, speedcheck, bad sectoring (810 only), block move, trace, manual, special print capabilites, disassembler, and more! — the best disk repair/editor at any price — and at the lowest price! **$28.95**

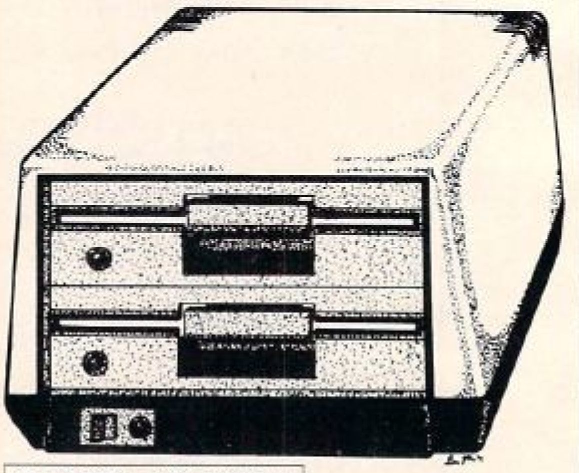
1st Class Postage Paid
California Residents add 6%, Foreign Orders add $2.50
C.O.D. add $2.00 — No credit cards
Prices subject to change
(213) 376-4105

ALLEN MACROWARE P.O. Box 2205 Redondo Beach, CA 90278

** Indicates Trademark of non-related company*

LOOK WHAT WE HAVE FOR YOUR ATARI® HOME COMPUTER

More Disk Drive for your money. In fact, with the Astra 1620, you get **two** superb Disk Drives for the price of one. The Astra 1620 is single or Double Density (software selectable) and completely compatible with Atari Dos or OSA+ Dos. When used as Double Density the Astra 1620 has the same capacity as Four Atari 810* Disk Drives.



The ASTRA 1620 DOUBLE DENSITY – DOUBLE DISK DRIVE HAS A SUGGESTED LIST PRICE OF $595.00.

ASTRA SYSTEMS

5230 CLARK AVENUE · LAKEWOOD, CA 90712 · PHONE: (213) 804-1475

* ATARI IS A REGISTERED TRADEMARK

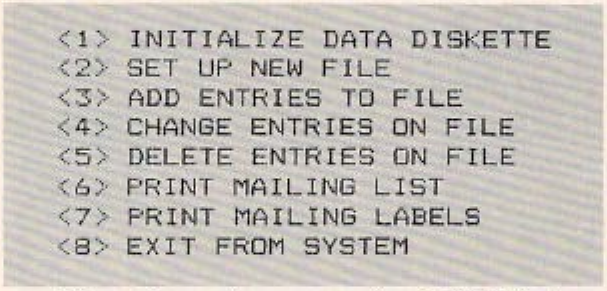
Make Your Apple User-Friendly

Karen Goeller McCullough

With an Apple II and a disk drive, you can use this versatile utility program to create menus that call other programs – or you can merge it with your own multifunction programs to create an effective master menu.

If you have an Apple II with Applesoft BASIC and at least one disk drive, this handy utility can save time and prevent confusion by generating menu programs. All you do is tell the program the number of options on your menu and their names. From that information a BASIC program is generated which presents a nicely formatted display of the options, allows the entry of a selection, and checks it for validity.

The figure shows a sample menu created using "Menu Generator." You simply add the code to tell the program where to go after the option has been selected. The Menu program can be used on its own to call other programs, or it can be merged into your own programs using the renumber and merge options in the "Renumber" program on the *System Master* diskette.



```
<1> INITIALIZE DATA DISKETTE
<2> SET UP NEW FILE
<3> ADD ENTRIES TO FILE
<4> CHANGE ENTRIES ON FILE
<5> DELETE ENTRIES ON FILE
<6> PRINT MAILING LIST
<7> PRINT MAILING LABELS
<8> EXIT FROM SYSTEM
```

Menu Generator uses an Apple DOS feature to create a program as a text file and then EXEC it. The EXEC command treats a text file as a series of commands that are executed just as though they had been entered from the keyboard. Delayed execution commands (those that have line numbers in front of them) are saved in memory to await a RUN command. (For more information on EXEC files, see pages 75-79 of the *DOS Manual*.) The EXEC command lets you write a BASIC program that will produce another BASIC program which can be immediately EXECed into memory and RUN, or SAVEd to disk as a program file. Menu Generator is an example of how this feature can be used for almost unlimited program generation.

Program Breakdown

In the Menu Generator program, lines 10-50 initialize the screen and variables. Line 100 sends us to line 1000 to begin processing. Lines 200 and 250 are subroutines that clear either a part of the screen (200) or a given line V (250). These are placed close to the beginning of the program to speed execution.

Lines 1000-1060 input the number of options desired on the menu. A string variable is used to input the number, and lines 1030-1040 scan the input string for valid numeric characters (ASCII 48-57). If an invalid character is detected, a flag (E) is set. The flag is then tested in line 1050, and, if true, execution is returned to the input statement at 1020.

This may seem a cumbersome process, since using a numeric variable would obviate the need for lines 1030-1050. However, in applications where an attractive screen format is important, this routine avoids the ?REENTER statement which appears if you try to enter a nonnumeric character into a numeric variable.

The options to appear on the menu are entered in lines 1070-1200. The array OP$ holds the option names, and the array element number also functions as the option number. For example, if option number 1 on the menu is to be INITIALIZE

DATA DISKS, then that will be the contents of OP$(1). After all the options are entered and checked for length of less than 30 characters, the program checks to see if changes are desired (1210-1420).

Beginning at line 2000, the text file which builds the program is created. The text file is opened in lines 2010-2040, and the first line is printed at 2045. Since the EXEC command itself does not clear the program currently in memory, the first line of the exec file issues an FP command, which prevents the EXECed program from overlaying the calling program. The POKE 34,24 in lines 2047 and 1420 prevents the screen from scrolling and the cursor from bouncing around while the EXEC file is being processed.

Menu Generator Variable List

A$	–yes or no answer input
CH$	–holds a single character for error checking
D$	–return + control-D (CHR$(13) + CHR$(4))
E	–error flag
H	–horizontal print location
I	–counter for FOR/NEXT loops
L	–length of longest option
L1	–temporarily holds length of each option
N	–number of options on menu
N$	–number of options (input string)
NN	–option number to change
NN$	–option number to change (input string)
N1	–option number selected on menu
OP$	–array holding option names
Q$	–quote mark (CHR$(34))
V	–vertical print location

Creating The New Program

The beginning of the new program being created (the menu program itself) is at line 2050. Lines 2050-2220 actually write the menu program, beginning with the header which will be lines 10-30 in the new program (lines 2050-2070 in the creating program). The variable N is set equal to the number of options, and the array OP$ is DIMed to N in line 40 (2080). The array OP$ is loaded with the option names in line 50 (2090), and line 2110 of the creating program causes the OP$ array to be printed as the DATA statement of line 70 of the new program. The length of the longest option line is found in line 2120; this information is used to calculate the horizontal positioning in line 2140. The same line also calculates the vertical positioning using the number of options (N).

After displaying the menu options, the program asks for the selection to be input. Input and validation procedures (2180-2220) are the same as those used for the option number input in the creating program. Line 2220 is the end of the delayed execution part of the text file, and it remains in memory while the EXEC function continues to the last two lines of the text file. Line 3010 causes the program which has been LOADed into memory from the text file to be SAVEd to disk as a program file called MENU-PROGRAM, and the next line causes it to be RUN.

Printing of the text file is concluded by line 3040, which CLOSEs the text file. The last line of the program issues the DOS EXEC command, which executes the text file. You now have the menu program SAVEd on disk and displayed on the screen, ready to make any modifications you might wish.

Menu Generator

```
1  REM   "***********************"
2  REM   "*   MENU GENERATOR    *"
6  REM   "***********************"
10  REM
20  TEXT : HOME : HTAB 13: PRINT "MENU
     GENERATOR"
30  VTAB 2: HTAB 1: FOR I = 1 TO 40: PRINT
     "-";: NEXT I
40  DIM OP$(12)
50 D$ =  CHR$ (13) +  CHR$ (4):Q$ =  CHR$
     (34): REM D$=CONTROL-D;Q$=QUOTE MA
     RK
100  GOTO 1000
199  REM  COMMONLY USED SUBROUTINES
200  VTAB 22: HTAB 1: CALL  - 956: VTAB
     22: RETURN
250  VTAB V: HTAB 1: CALL  - 868: VTAB
     V: RETURN
999  REM
1000  REM  ENTER NUMBER OF OPTIONS DESI
     RED ON THE MENU
1001  REM
1010  GOSUB 200: HTAB 1: PRINT "YOU MAY
      CHOOSE UP TO 12 MENU OPTIONS OF":
      PRINT "UP TO 30 CHARACTERS EACH I
     N LENGTH"
1020 V = 5: GOSUB 250: INPUT "ENTER NUM
     BER OF MENU OPTIONS (1-12) ";N$
1030 E = 0: FOR I = 1 TO  LEN (N$):CH$ =
      MID$ (N$,I,1):CH =  ASC (CH$): IF
     CH < 48 OR CH > 57 THEN E = 1
1040  NEXT I
1050  ON E GOTO 1060,1020
1054  REM
1055  REM  ENTER THE MENU OPTIONS
1056  REM
1060 N =  VAL (N$): IF N < 1 OR N > 12 THEN
      GOTO 1020
1070 V = 7: GOSUB 250: PRINT "ENTER OPT
     ION NAME NEXT TO THE NUMBER"
1075  GOSUB 200: PRINT "NO COMMAS, COLO
     NS OR QUOTE MARKS IN THE": PRINT "
     MENU OPTIONS PLEASE"
1080  FOR I = 1 TO N
1090 V = 7 + I: GOSUB 250: PRINT I;: INPUT
     " ";OP$(I)
1100  IF  LEN (OP$(I)) > 30 THEN  GOTO
     1090
1200  NEXT I
1210 V = 20: GOSUB 250: PRINT  TAB( 10)
     "ANY CHANGES (Y/N) ";: GET A$: IF
     A$ = "N" THEN  GOTO 1400
1220  IF A$ <  > "Y" THEN  GOTO 1210
1230 V = 20: GOSUB 250: PRINT  TAB( 5)"
     CHANGE NO. OF OPTIONS (Y/N) ";: GET
     A$: IF A$ = "N" THEN  GOTO 1300
1240  IF A$ <  > "Y" THEN  GOTO 1230
1250  GOTO 1000
1300 V = 20: GOSUB 250: HTAB 5: INPUT "
     ENTER OPTION NUMBER TO CHANGE ";NN
     $: IF  LEN (NN$) > 2 THEN  GOTO 13
     00
1310 E = 0: FOR I = 1 TO  LEN (NN$):CH$
      =  MID$ (NN$,I,1): IF  ASC (CH$) <
     48 OR  ASC (CH$) > 57 THEN E = 1
1320  NEXT I
1330  ON E GOTO 1340,1300
1340 NN =  VAL (NN$): IF NN < 1 OR NN >
     N THEN  GOTO 1300
1350 V = 7 + NN: GOSUB 250: PRINT NN;: INP
     UT" ";OP$(NN)
1360  IF  LEN (OP$(NN)) > 30 THEN  GOTO
     1350
1370  GOTO 1210
1400  FOR I = 1 TO N:L1 =  LEN (OP$(I))
     : IF L1 > L THEN L = L1
1410  NEXT I
1420  POKE 34,24
2000  REM  BUILD TEXT FILE
2010  PRINT D$;"OPEN MENU-FILE"
2020  PRINT D$;"DELETE MENU-FILE"
2030  PRINT D$;"OPEN MENU-FILE"
2040  PRINT D$;"WRITE MENU-FILE"
2045  PRINT "FP"
2047  PRINT "POKE 34,24"
2050  PRINT "10 REM MENU PROGRAM"
2053  PRINT "12 TEXT:HOME"
2055  PRINT "15 VTAB 1:FOR I=1 TO 40:PR
     INT "Q$"-"Q$";:NEXT I"
2060  PRINT "20 VTAB 2:PRINT TAB(18) "Q
     $"MENU"Q$
2070  PRINT "30 VTAB 3:FOR I=1 TO 40:PR
     INT "Q$"-"Q$";:NEXT I"
2080  PRINT "40 N=";N;":DIM OP$(";N;")"
2090  PRINT "50 FOR I=1 TO N:READ OP$(I
     ):NEXT I"
2100  PRINT "60 GOTO 80"
2110  PRINT "70 DATA ";: FOR I = 1 TO N
     - 1: PRINT OP$(I);",";: NEXT I: PRINT
     OP$(N)
2120  PRINT "80 FOR I=1 TO N:L1=LEN(OP$
     (I)):IF L1>L THEN L=L1"
2130  PRINT "90 NEXT I"
2140  PRINT "100 V=(INT(24-N)/2)+1:H=IN
     T((40-(L+4))/2)"
2160  PRINT "110 IF N<=9 THEN FOR I=1 T
     O N:VTAB V+I:PRINT TAB(H) "Q$"<"Q$
     "I"Q$"> "Q$";OP$(I):NEXT:GOTO 120"
2170  PRINT "112 FOR I=1 TO 9:VTAB V+I:
     PRINT TAB(H) "Q$"<"Q$"I"Q$"> "Q$";
     OP$(I):NEXT I"
2175  PRINT "114 FOR I=10 TO N:VTAB V+I
     :PRINT TAB(H-1) "Q$"<"Q$"I"Q$"> "Q
     $";OP$(I):NEXT I"
2180  PRINT "120 VTAB 23:HTAB 8: INPUT
     "Q$"ENTER SELECTION : "Q$";N$"
2190  PRINT "130 IF LEN(N$)>2 OR LEN(N$
     )<1 THEN GOTO 120"
2200  PRINT "140 E=0:FOR I=1 TO LEN(N$)
     :CH$=MID$(N$,I,1):IF ASC(CH$)<48 O
     R ASC(CH$)>59 THEN E=1"
2210  PRINT "150 NEXT I:IF E=1 THEN GOT
     O 120"
2220  PRINT "160 N1=VAL(N$):IF N1<1 OR
     N1>N THEN GOTO 120"
3010  PRINT "SAVE MENU-PROGRAM"
3030  PRINT "RUN"
3040  PRINT D$;"CLOSE"
3050  PRINT D$;"EXEC MENU-FILE"
```

©

Computer Case Company

WE MAKE A GREAT CASE FOR YOUR COMPUTER.

One size does not fit all. Our cases are designed for specific hardware configurations. When you put your computer in our case, it fits hand-in-glove. Once your equipment is safely inside the attache-style carrying case, it never has to be taken out again. To operate, simply remove the lid and connect the power. To store your computer, disconnect the power, enclose your disks, working papers, and manuals in the compartment provided, and attach the lid. It's as easy as that.

IB502

IB501

● AP101	Apple II with Single Drive	$109
● AP102	Apple II with Two Disk Drives	119
● AP103	Apple II 9-inch Monitor & Two Drives	129
● AP104	Apple III, Two Drives & Silentype Printer	139
● AP105	13" Black & White Monitor with Accessories	99
● AP106	Amdek Color I, II or III Monitor	119
● FR152	Franklin Ace 1000 or 1200 with Two Drives	119
● FR153	Franklin Ace 1000 or 1200 with Two Drives & 9" Monitor	139
● RS201	TRS-80 Model I Computer, Expansion Unit & Drives	109
● RS204	TRS-80 Model III	129
● AT301	ATARI 400 or 800 Computers with Peripherals	109
● P401	Paper Tiger Printer (400/445/460)	99
● P402	Centronics 730/737 & Radio Shack Printer	89
● P403	Epson MX70 or MX80, Microline 82A Printer or Color Computer	89
● P404	Epson MX100 Printer	99
● P405	IDS 560 or Prism 132 Printer	109
● P406	C. Itoh Starwriter/Printmaster F-10 Printer	119
● P407	Okidata Microline 83A or 84 Printer	99
● P408	C. Itoh Prowriter 2 Printer	99
● P409	C. Itoh Prowriter (Apple Dot Matrix) or NEC PC8023 Printer	89
● IB501	IBM Personal Computer with Keyboard	129
● IB502	IBM Monochrome Monitor	99
● HP601	HP41 with Accessories	99
● CM702	Commodore 64 (or Vic 20) with One Drive	119
● CM703	Commodore Model 64 with Two Drives	129
● CM704	Commodore Model 64 with Dataset	109
● NS010	North Star Advantage	139
● CC80	Matching Attache Case (5")	85
● CC90	Matching Attache Case (3")	75
● CC91	Matching Accessories Case (5¼" Diskettes, Paper, etc.)	95
● CC92	5.25" Diskette Case (Holds 75 Diskettes)	49
● CC50	Case Cart	79

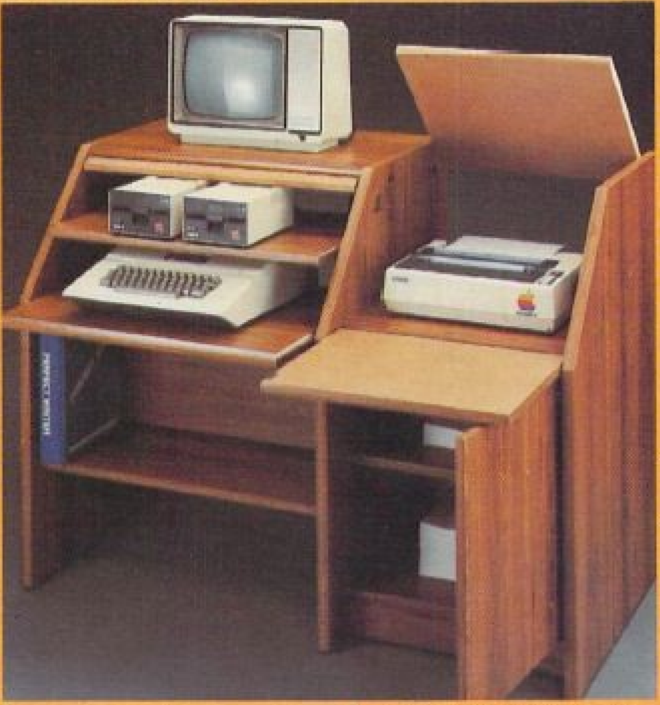
CALL TOLL FREE: (800) 848-7548

Computer Case Company, 5650 Indian Mound Court, Columbus, Ohio 43213 (614) 868-9464

THERE'S A COMPUTER BORN EVERY MINUTE... GIVE IT A HOME.

For **$89.95** with the CS-1632 you can house your computer, peripherals, and accessories without spending a fortune.

For those with a large computer family the CS-2748 gives you all the room you need for your computer, monitor, printer, peripherals, software, etc. at a price that's hard to believe: **$299.95.**



The CS-1632 computer storage cabinets compact yet functional design fits almost anywhere while housing your computer monitor, joysticks, software, books and peripherals all for only $89.95.
The slide out shelf puts the computer at the right height and position for easy comfortable operation.
The fold up locking door keeps unwanted fingers off the key board when not in use.
To store joysticks just turn them upside down and slide them into the inverted storage rack.
Twist tabs on the back of center panel allow for neat concealed grouping of wires, while power packs rest hidden behind center panel on shelf.
The slide out software tray has room for 14 cartridges or cassettes and up to 30 diskettes. Most brands of software will fit between the adjustable partitions with a convenient hook for the spare key at rear.
Stand fits Atari 400 & 800, Commodore 64 & VIC 20, TI 99/4A and TRS-80.
Cabinet dimensions overall 36" high x 33-7/8" wide x 16" deep.

To order CS-1632 send $89.95 to: To order CS-2748 send $299.95 to:

HYTEC SYSTEMS
P. O. Box 446
West Lynn, OR 97068
For Fast Phone Orders Call Toll Free **1-800-547-3100**
Inside Oregon Call **(503) 635-6667**

Name ____________________
Address ____________________
City ____________ State ________ Zip ________
Quantity ______ CS-1632 Quantity ______ CS-2748
☐ Golden Oak Finish ☐ Natural walnut finish
☐ My personal check, cashiers check or money order is enclosed.
☐ Bill my VISA # ____________ Exp. Date ____
☐ Bill my MasterCard # ____________ Exp. Date ____
☐ Please include freight charge on my VISA or MasterCard.
Card Holders Signature ____________________

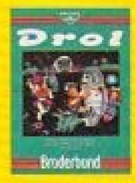
Immediate shipment if in stock. If not, allow 3-4 weeks for delivery. If personal check is sent allow additional 2 weeks. CS-1632 ships UPS freight collect from Oregon. CS-2748 ships by truck freight collect from Oregon. Prices subject to change. Shipment subject to availability.

Both the CS-1632 and CS-2748 ship unassembled in two cartons. Assembly requires only a screwdriver, hammer, and a few minutes of your time.
Choice in simulated woodgrain of warm golden oak or rich natural walnut finish.

The two slide-out shelves put the keyboard at the proper operating height while allowing easy access to the disk drives.
The bronze tempered glass door protecting the keyboard and disk drives simply lifts up and slides back out of the way during use.
Twist tabs on the back of the center panel allow for neat concealed grouping of wires while a convenient storage shelf for books or other items lies below.
The printer sits behind a fold down door that provides a work surface for papers or books while using the keyboard. The lift up top allows easy access to the top and rear of the printer. A slot in the printer shelf allows for center as well as rear feed printers.
Behind the lower door are a top shelf for paper, feeding the printer, and a bottom shelf to receive printer copy as well as additional storage.
Stand fits same computers as the CS-1632 as well as the Apple I and II, IBM-PC, Franklin and many others.
The cabinet dimensions overall: 39-1/2" high x 49" wide x 27" deep.
Keyboard shelf 20" deep x 26" wide. Disk drive shelf 15-34" deep x 26" wide. Top shelf for monitor 17" deep x 27" wide. Printer shelf 22" deep x 19" wide.

H HYTEC SYSTEMS

Bet'cha Can't Play Just One!

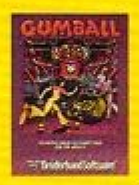


Drol™

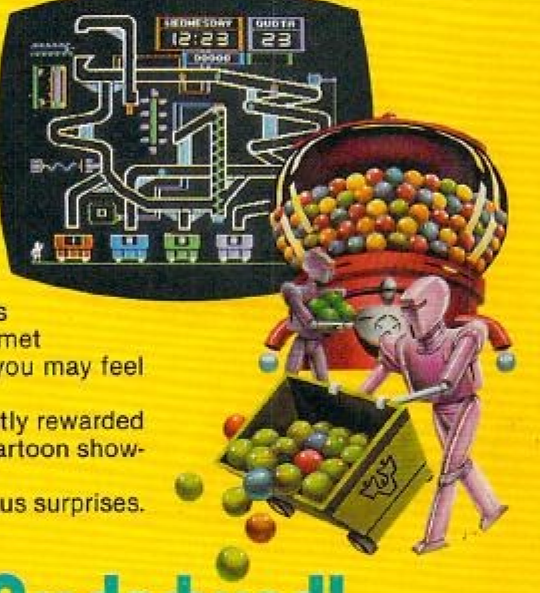
You will soon come to expect the unexpected in the hilarious and challenging underground dream world of Drol. A little red-headed girl and her propeller-beanied brother have been lured by a witch doctor's curse into the multi-leveled ruins of a lost civilization. It's your task — as a hero equipped with a rocket backpack and full-screen radar scope — to dodge hopping scorpions, monsters and snakes, flying turkeys, swords, daggers, arrows, magnets, witch doctors, and vacuum cleaners(!) in your attempts to rescue the children and reunite them with their mother. Each new level of game play is full of surprises.

Drol's wry sense of humor and amazingly detailed cartoon imagery, make this game a charmer!

For the Apple II/II + /IIe, Atari, and Commodore 64 home computers in disk format.



Gumball™



Hours of fun await you at the Sticky Sole Gumball Factory — where you'll be working against the clock to sort a tasty collection of colorful gumballs.

Your job may seem sweet at first, but after you've discovered the explosive-laced gumballs (placed by over-zealous dental assistants) or met your irritating supervisor (who is eager to undo your best efforts), you may feel that you have bitten off more than you can chew.

If, against all odds, you meet your day's quota, you'll be promptly rewarded with a promotion (to a more challenging position) and an amusing cartoon showing your higher standard of living.

Gumball — a new fast action game filled with colorful and delicious surprises.

For the Apple II/II + /IIe.

Irresistible Fun From Brøderbund!

Brøderbund Software™

Discover the Difference

17 PAUL DRIVE, SAN RAFAEL, CALIFORNIA 94903

Apple II/II + /IIe, Atari, and Commodore 64 are trademarks of Apple Computer, Inc., Atari, Inc., Commodore Electronics Ltd., respectively.

ur creativity.

Be quick on the draw. PictureWriter is magic! Create any shape or pattern, instantly. Fill areas with glowing colors and even hear pictures set to music.

PictureWriter brings out the artist in anyone. With this program, your child can create his or her own picture gallery and watch the computer redraw the pictures like magic on the screen. PictureWriter also includes a library of masterpieces by other "picture writers" that can be colored, edited and redrawn.

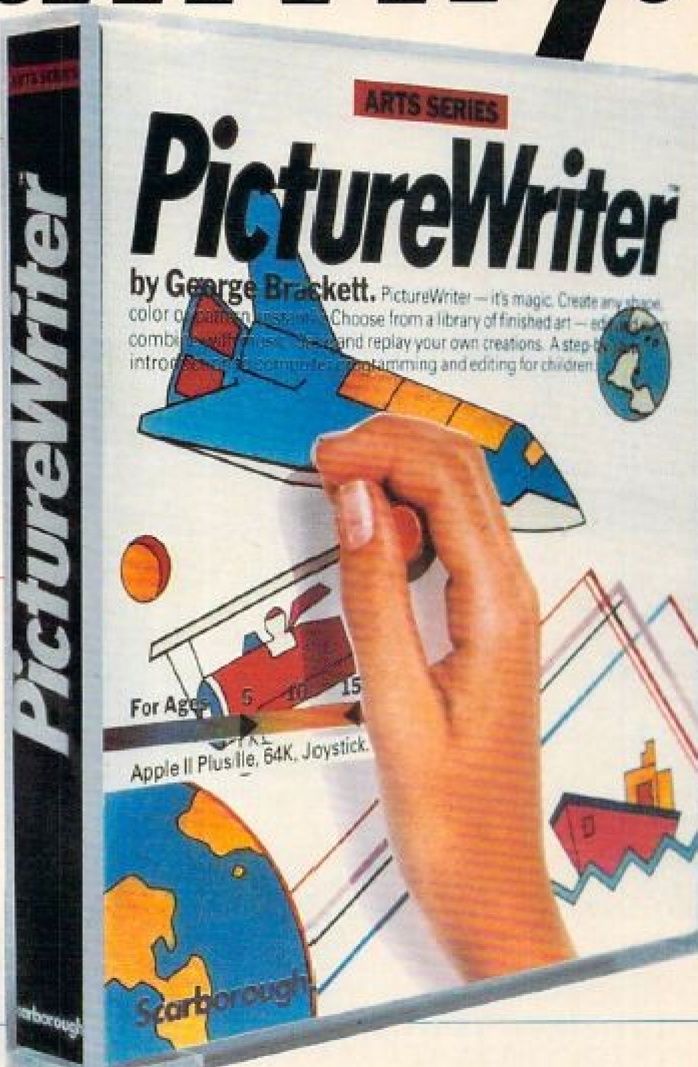
Like all Scarborough programs, PictureWriter encourages experimentation and continually challenges the child to explore new avenues. And all the while, PictureWriter subtly develops the child's familiarity with the fundamentals of step by step computer programming.

Getting started is simple. The built-in tutorial zips the artist into the program quickly and keeps him or her creatively occupied for hours.

The possibilities are endless with PictureWriter. In fact, children find it so captivating that parents will probably want to doodle with it, too. And why not?

You can't stay an adult forever.

Available for Apple® $39.95
(Soon, Atari®)



Reproduced on Wabash disks.

You'll grow with us.

ough System.

WE'LL BACK YOU UP!

"The best back-up and disk utilities to date."

DEALER AND DISTRIBUTOR INQUIRIES INVITED

Order from
MICRO-WARE DIST. INC.
1342B Rt 23 Butler, NJ 07405
201-838-9027

MasterCard

VISA

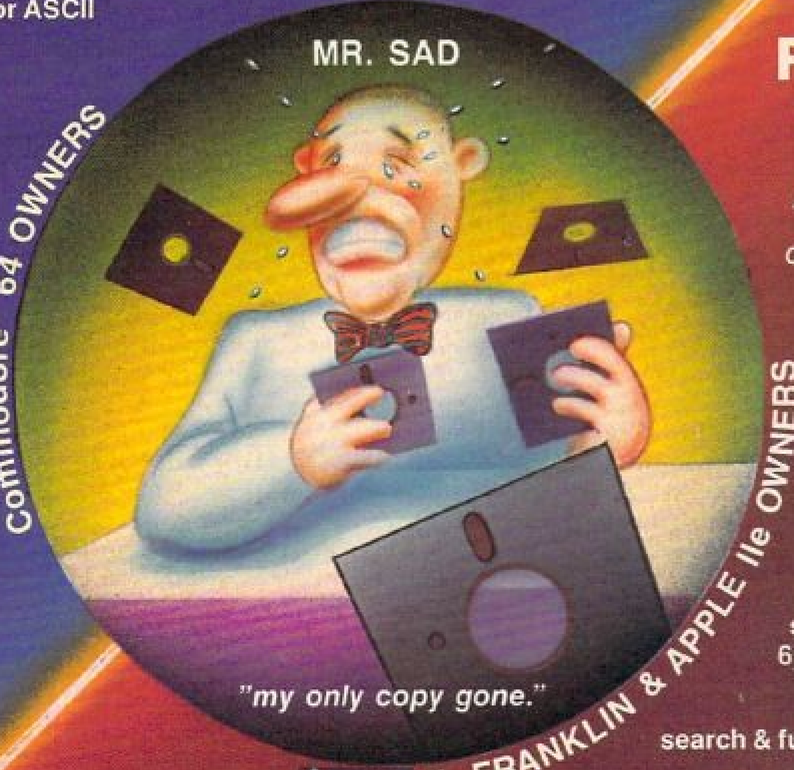
THE CLONE MACHINE™ *FROM MICRO-WARE DIST. INC.*

Take control of your 1541 disk drive with this indispensible disk users tool. The Clone Machine will allow you to copy programs, files, full or partial disks, and even allow track/block editing.
Package includes:

1) Complete and thorough users manual
2) Copy with one or two drives
3) Copy all file types including relative files
4) Investigate and back-up many protected disks.
5) View track/block in HEX or ASCII
6) Easily edit track blocks
7) Display full contents of directory and print
8) Change program names, add, delete files with simple keystroke
9) Easy disk intitialization
10) Supports up to four drives

All this for only $49.95

Special limited introductory offer
$39.95



The NEW REVISED Nibbles Away II™

version C by COMPUTERapplications

1) Full Apple IIe compatibility
2) New auto-loading to simplify parameter access
3) Added printer compatibility
4) Enhanced printing formats
5) CTRL P screen snapshot to printer
6) Enhanced sector editing
7) New disk data search & full disk diagnostics
8) Subscription to NIBBLE NEWS Available for back up hints and new parameter settings.

NIBBLES AWAY II version C is still the best and most supported back up program available. Written about in the New York Times Business Section, Science 83, Digital retailing, and other publications as one of the most popular of its kind. A necessary program for all Apple & Franklin owners. List $69.95

Limited special offer only **$64.95**

Commas And Colons In Applesoft Strings: An Easy Way To Use Them

Donald W. Watson

Commas and colons are not allowed with Applesoft strings — and this can be troublesome at times. Here's a solution. Also included is a program for Apple II disk users.

The Keyboard Problem

INPUT X$ is the convenient instruction for entering strings with an Applesoft II BASIC program; however, the string to be entered under the variable name X$ may not contain commas or colons. If either is present, the string will be truncated at the first occurrence when the RETURN key is pressed. The comma or colon and all characters following will be lost, and Applesoft will send the ?EXTRA IGNORED message to the printer or to the screen.

In programs written for business use, it is often essential to include commas and colons in strings entered by the user. Programmers may not mind, but consider the user's frustration on learning that he or she cannot use commas or colons in places where they are normally required for acceptable format. For example, JONES, JAMES. J. is a common format for names in a list; RECEIPTS: might be a desirable heading for a list or group on a business report or ledger. In the latter example, the colon can be avoided by underlining the heading, but only at the expense of the user's choice, printer time, and perhaps report line space. Restricting alternatives is not in the user's interest. Here is a practical solution to the problem.

A Keyboard Solution

The *Applesoft BASIC Programming Reference Manual* is not much help on this subject although a clue to a solution is offered in Chapter 6 where the INPUT and GET instructions are defined and discussed. On page 68, a suggestion is made that "serious programmers GET numbers" by using a GET X$ instruction, where the keyboard response will be a string assigned to the string variable X$ when the RETURN key is pressed.

"String Entry" allows the entry of strings which can contain *all* characters from the Apple II keyboard. But String Entry does much more. The program contains routines which duplicate the most important Apple II string-editing capabilities (right- and left-arrow functions). It also provides some useful entry control functions for convenience in writing, displaying, and deleting strings.

A Free Keyboard

Type the listing into memory and proofread it carefully. When you're sure it is correct, SAVE it to a disk with a short name like STRENT. Then type RUN (with the program still in memory). The instruction line will appear. Experiment with the string entry process, noting that you now have the full freedom of the keyboard. You can enter strings with any characters you like, and you have normal editing functions with entry and deletion control. Best of all, the ?EXTRA IGNORED message never appears, and nothing is ignored unless you choose to have it ignored.

Most of String Entry (it's about 600 bytes long) can be used, with slight modification, in a larger program. If used to control string entry for more than one or two fields, it must be generalized for use as a subroutine, mostly by using integer variables V% and H% in the calling routine. VTAB V% and HTAB H% instructions can then be used in the subroutine to allow complete freedom when choosing a location for the string display on the screen.

The Apple II Disk Problem

The keyboard problem with commas and colons to be used in strings has been solved by avoiding the INPUT X$ instruction and using a GET X$ routine instead. But Apple II disk operations require the use of the INPUT X$ instruction to retrieve string data from a disk text file. If the string to be retrieved contains commas or colons, the ?EXTRA IGNORED message will occur; the string will be truncated as if it were entered from the keyboard in response to INPUT X$.

To correct this, try these two simple changes and some short additions to the String Entry program.

1. Delete: GOTO 1020 from the end of line 1190.

2. Add the lines below to the String Entry program.

3. SAVE the modified and expanded program String Entry under its abbreviated name, STRENT.

```
1300  REM WRITE S$ CONTENT TO DISK
1310  PRINT D$;"OPEN STRFILE"
1320  PRINT D$;"DELETE STRFILE"
1330  PRINT D$;"OPEN STRFILE"
1340  PRINT D$;"WRITE STRFILE"
1350  PRINT S$
1360  PRINT D$;"CLOSE STRFILE"
1400  REM RETRIEVE S$ CONTENT FROM DISK
1410 S$ = ""
1420  PRINT D$;"OPEN STRFILE"
1430  PRINT D$;"READ STRFILE"
1440  INPUT S$
1450  PRINT D$;"CLOSE STRFILE"
1500  REM DISPLAY RETRIEVED S$ CONTENT
1510  VTAB 20: HTAB 8: PRINT  TAB( 39);
     : HTAB 9: PRINT S$: GOTO 1020
```

Save this expanded version to disk under the original filename STRENT.

Type RUN to execute the expanded program still in memory. The operator instruction line will appear. Using no commas and no colons, experiment with a few string entries. Each string entered will be stored on disk, and the program will echo the string by displaying it (as retrieved from the disk text file) a second time.

Now, perform a test. Enter a string containing a comma or colon, or both. Try NAME: JONES, JAMES J., for instance. When you have entered the string, it remains displayed at the string entry format line. It goes to the STRFILE at the disk under the permanent variable name S$. S$ in computer memory is nulled, S$ is retrieved from STRFILE, and the retrieved content of S$ is displayed on the screen.

But disaster strikes again. First, the dreaded ?EXTRA IGNORED message is displayed, and then the string is displayed in incomplete form. Read on for help.

An Apple II Disk Solution

The Apple II disk system (DOS 3.2 or DOS 3.3) will accept the contents of S$ as a literal string if the contents begin with a quote (") mark. The disk retrieval problem can be avoided by changing S$ temporarily with the statement S$=CHR$(34)+S$.

To try this technique, just change line 1350 to the following:

```
1350 PRINT CHR$(34) + S$
```

SAVE the program once more under the filename STRENT and RUN it. Now, you will find that the test string NAME: JONES, JAMES J. can be correctly entered and correctly retrieved. And so can any string containing any characters from the Apple II keyboard, including commas and colons.

String Entry

```
1000  REM STRING ENTRY
1010  HOME : DIM C$(30):D$ =  CHR$ (4)
1020  VTAB 15: HTAB 9: PRINT "TYPE ";
1025  INVERSE : PRINT "E";: NORMAL : PRINT
     " TO ENTER NEW STRING ";
1030  GET E$: VTAB 15: HTAB 9: PRINT  TAB(
     39)
1040  VTAB 10: HTAB 8: PRINT "?";: FOR
     X = 1 TO 25:C$(X) = "": PRINT "-";
     : NEXT X: HTAB 9:X = 1
1050  IF X > 25 THEN  PRINT  CHR$ (7): GOTO
     1160
1060  GET C$: IF X > 1 THEN 1090
1070  IF  ASC (C$) = 13 THEN 1190
1080  IF  ASC (C$) < 33 OR  ASC (C$) >
     90 THEN S$ = "": GOTO 1040
1090  IF C$(1) = "0" AND X = 2 AND  ASC
     (C$) = 13 THEN S$ = "": GOTO 1190
1100  IF  ASC (C$) = 13 THEN 1160
1110  IF  ASC (C$) > 31 AND  ASC (C$) <
     91 THEN  PRINT C$;:C$(X) = C$:X =
     X + 1: GOTO 1050
1120  IF  ASC (C$) = 8 THEN X = X - 1: HTAB
     (8 + X): GOTO 1060
1130  IF  ASC (C$) = 21 AND C$(X) <  >
     "" THEN X = X + 1: HTAB (8 + X): GOTO
     1050
1140  IF  ASC (C$) = 21 THEN  HTAB (8 +
     X): GOTO 1060
1150  HTAB (8 + X): GOTO 1040
1160 ST$ = "": FOR L = 1 TO X - 1:ST$ =
     ST$ + C$(L): NEXT L
1170 R$ =  RIGHT$ (ST$,1): IF  ASC (R$)
     = 32 THEN ST$ =  LEFT$ (ST$, LEN
     (ST$) - 1): GOTO 1170
1180 S$ = ST$
1190  VTAB 10: HTAB 8: PRINT  TAB( 39);
     : HTAB 9: PRINT S$: GOTO 1020
```
©

Use the handy reader service cards in the back of the magazine for information on products advertised in

COMPUTE!

Apple Disk Drive

The Half Track disk drive from Wholesale Technology is a 5¼ inch disk drive that stands half as high as a standard Apple drive. It provides 160K of double density storage and is compatible with Apple II, II+, and IIe computers running DOS 3.2 or 3.3.

The drive features auto-eject of diskettes, a quick-release controller cable, and a 12 millisecond head access time.

Drive 1, complete with filer DOS 3.3, controller, cable, and documentation, sells for $399.95.

Wholesale Technology, Inc.
1530 South Sinclair
Anaheim, CA 92806
(714) 978-9820

Wholesale Technology's disk drive is only 41 millimeters high and provides 160K bytes of storage.

ComputAbility™ ATARI

THE LARGEST ATARI MAIL ORDER COMPANY PRESENTS THE HOTTEST TITLES

TI – CARTRIDGE	
PAC-MAN	34.95
CENTIPEDE	34.95
DEFENDER	34.95
DIG DUG	34.95
DONKEY KONG	34.95
VIC-20 – CARTRIDGE	
PAC-MAN	34.95
CENTIPEDE	34.95
DEFENDER	34.95
DIG DUG	34.95
DONKEY KONG	34.95
STARGATE	34.95
ROBOTRON	34.95
COMMODORE 64 – CARTRIDGE	
PAC-MAN	34.95
CENTIPEDE	34.95
DEFENDER	34.95
DIG DUG	34.95
DONKEY KONG	34.95
STARGATE	34.95
ROBOTRON	34.95
APPLE – DISK	
PAC-MAN	27.95
CENTIPEDE	27.95
DEFENDER	27.95
DIG DUG	27.95
DONKEY KONG	27.95
STARGATE	27.95

QUANTITIES LIMITED S + H $2.00 PER ORDER

(800) 558-0003
IN WISCONSIN
(414) 351-2007
ComputAbility • P.O. Box 17882 • Milwaukee, WI 53217

try your skills at
TRAIN DISPATCHER

REALISTIC SIMULATION OF A COMPUTER CENTRALIZED TRAFFIC CONTROL (CTC) OFFICE. HOW QUICKLY AND EFFICIENTLY CAN YOU MANEUVER TRAINS OVER A 20 STATION, 150 MILE TERRITORY? BUILD YOUR OWN SKILLS FROM "CUB DISPATCHER" TO "TRAIN MASTER".

24 DISPLAYS	including Train Sheets, Block Permits, Territory Overview, 20 interlockings.
FULL CONTROL	Throw switches, clear and cancel signals, route and maneuver up to 12 trains moving in both directions.
REAL TIME	All displays continuously updated, including train location, switch positions, signals, block permits, and crew time.
AUTHENTIC	Created by designers of CTC Systems for operating railroads.

TRAIN DISPATCHER is an action game, that provides a dramatic and realistic picture of true-to-life railroad operations.

Commodore 64 and VIC 20 are registered Trademarks of Commodore Business Machines, Inc.
ATARI: A Warner Communications Company   Apple is registered trademark of Apple Computer, Inc.

SIGNAL COMPUTER CONSULTANTS, LTD.
470 Streets Run Road Pittsburgh, PA 15236

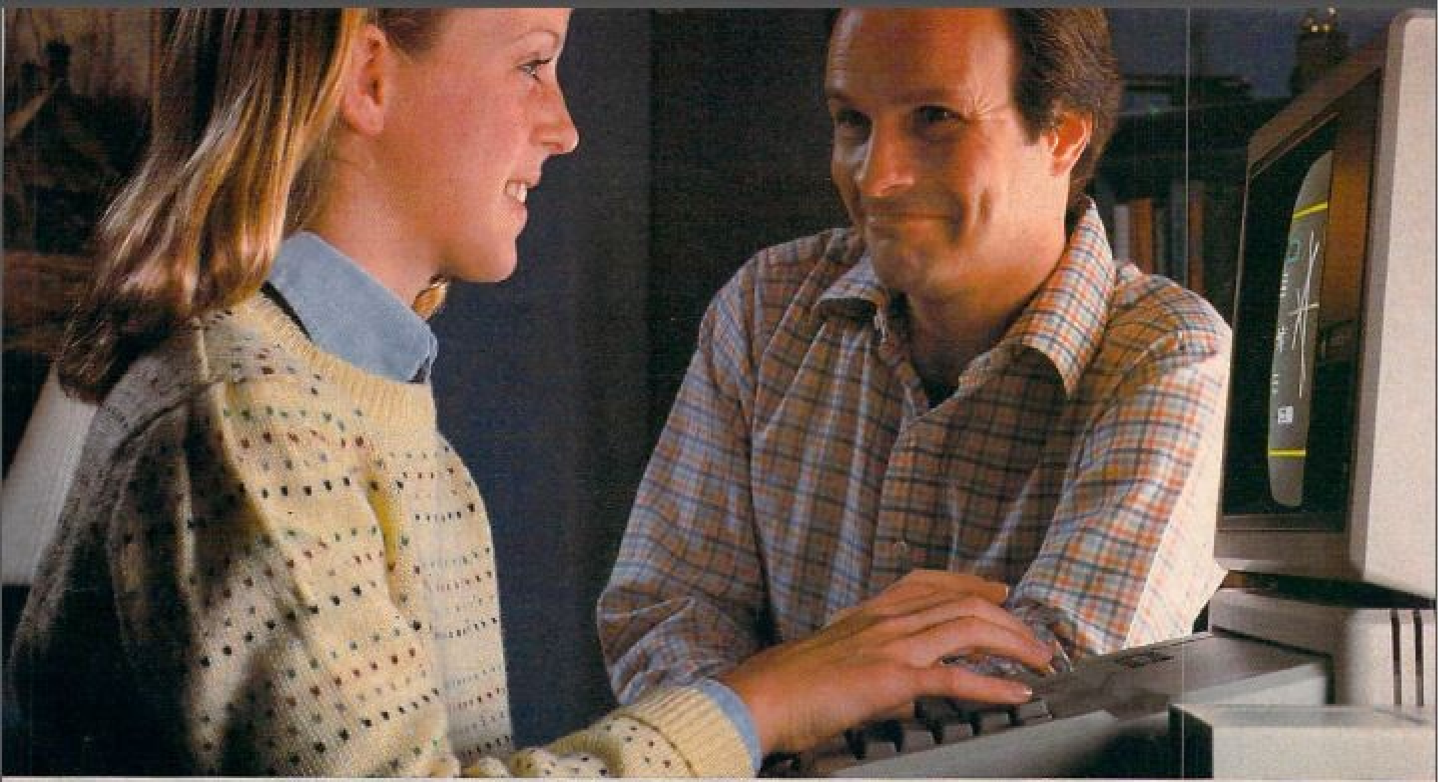
Enclosed is $24.95 for Train Dispatcher + $2.50 for postage and handling. PA residents add state sales tax.

Charge To: ☐ Master Card ☐ VISA Card # ________ Exp. Date ________

CHECK ONE
☐ VIC 20™, Tape* ☐ Commodore™ 64 Tape ☐ Commodore™ 64 Disk
☐ Atari® 400/800 Tape** ☐ Apple II®, Disk (29.95)
☐ Manual is included with game or to be purchased separately at $4.00.

*VIC 20 requires 16K/**Atari 400/800 requires basic

Signature ________
Name ________
Street ________
City ________ State, Zip ________



GIVE THE GIFT THAT MAKES YOUR APPLE SHINE.

This Christmas give PLATO® educational courseware and start your family on an exciting learning adventure.

Discover the difference quality courseware makes. Begin with the new PLATO Computer Concepts* series: The Computer Keyboard, Storage and Memory, Files and Editing, and Databases. Put these lessons into practice along with Keyboarding† and Computer Literacy to help your family really understand the computer.

Widen your child's world with these other PLATO lessons.

Grade school kids can have fun while they learn Basic Number Facts, Whole Numbers, Decimals, and Fractions. For the teen-ager in your family there are PLATO lessons in Elementary Algebra‡, Physics-Elementary Mechanics, and Foreign Languages. All are part of a growing library of quality educational programs.

Ask for PLATO at selected retail outlets.

PLATO courseware for microcomputers is available for the Apple II Plus and Apple IIe. Selected lessons are also available for the TI99/4A and Atari 800. For a free PLATO catalog: Call toll-free 800/233-3784. (In Calif., call 800/233-3785.) Or write Control Data Publishing Co., P.O. Box 261127, San Diego, CA 92126.

*Developed with Continuous Learning Corporation
†Developed with Gregg/McGraw-Hill
‡Developed with Courses by Computers, Inc.

Warranty available free from Control Data Publishing Co., 4455 Eastgate Mall, San Diego, CA 92121

PLATO
COMPUTER-BASED EDUCATION

CONTROL DATA
PUBLISHING

Flight Simulator II

For Apple, Atari, & Commodore 64™

Put yourself in the pilot's seat of a Piper 181 Cherokee Archer for an awe-inspiring flight over realistic scenery from New York to Los Angeles. High speed color-filled 3D graphics will give you a beautiful panoramic view as you practice takeoffs, landings, and aerobatics. Complete documentation will get you airborne quickly even if you've never flown before. When you think you're ready, you can play the World War I Ace aerial battle game. Flight Simulator II features include ■ animated color 3D graphics ■ day, dusk, and night flying modes ■ over 80 airports in four scenery areas: New York, Chicago, Los Angeles, Seattle, with additional scenery areas available ■ user-variable weather, from clear blue skies to grey cloudy conditions ■ complete flight instrumentation ■ VOR, ILS, ADF, and DME radio equipped ■ navigation facilities and course plotting ■ World War I Ace aerial battle game ■ complete information manual and flight handbook.

See your dealer . . .

or write or call for more information. For direct orders please add $1.50 for shipping and specify UPS or first class mail delivery. American Express, Diner's Club, MasterCard, and Visa accepted.

Order Line: 800/637-4983

subLOGIC
Corporation
713 Edgebrook Drive
Champaign IL 61820
(217) 359-8482 Telex: 206995

QUESTRON

Live the Fantasy and the Adventure.

STRATEGIC SIMULATIONS INC. PRESENTS A FANTASY ADVENTURE GAME: QUESTRON™
ONE OF THE FINEST CHAPTERS IN THE NEVER-ENDING SAGA OF THE BATTLE BETWEEN GOOD AND EVIL
Starring YOU as THE HERO • MESRON, THE GOOD WIZARD • MANTOR, THE EVIL SORCEROR
AND HIS HORDES OF HERO-CRUNCHING MONSTERS • Written and directed by CHARLES DOUGHERTY
ON 48K DISK FOR YOUR APPLE® II WITH APPLESOFT ROM CARD, APPLE II+, IIe, OR APPLE III. ALSO FOR ATARI® HOME COMPUTERS.

APPLE® version now showing at a computer/software or game store near you. ATARI® version coming Spring 1984.

SSI

PG THIS GAME RATED POSITIVELY GREAT. Ideal for Fantasy Adventurers of all ages.

APPLE is a registered trademark of Apple Computer, Inc. **STRATEGIC SIMULATIONS INC** ATARI is a registered trademark of Atari, Inc.

If there are no convenient stores near you, VISA & Mastercard holders can order direct by calling **800-227-1617, ext. 335** (toll free). In California, call 800-772-3545, ext. 335. QUESTRON™ goes for $49.95, plus $2.00 for shipping and handling.

To order by mail, send your check to: STRATEGIC SIMULATIONS INC, 883 Stierlin Road, Bldg. A-200, Mountain View, CA 94043. (California residents, add 6.5% sales tax.) All our games carry a "14-day satisfaction or your money back" guarantee.

WRITE FOR A FREE COLOR CATALOG OF ALL OUR GAMES.