# Commodore EXEC

Edwin King

*There are times when you'd benefit from being able to access subroutines directly from a disk file. They're not in your program (taking up space), but they can be accessed from a main program, executed, and then the main program continues. Called the EXEC command on those few versions of BASIC which have it, this technique is worth adding to your programmer's bag of tricks. For all Commodore machines. We'll go through the process step by step so you can try the technique and watch it in operation.*

Here's a way to store all of your favorite subroutines on disk and have programs call them when they're needed, without having to retype or append or use up memory space.

## The Technique

The idea behind the EXEC command (as found in Applesoft BASIC; Commodore Microsoft has no such thing) is to execute a subroutine from disk as if it were typed directly into the computer. Just call a command from the disk, in the form of a character string, and start POKEing to the *dynamic keyboard*.

For those not familiar with the dynamic keyboard concept, let me review. Every time a key is pressed, the computer stores the ASCII code representation of it in a place called the "keyboard buffer." It keeps doing this until you press RETURN (which also goes into the buffer), then it goes back to evaluate and execute what you just typed in. Lest the computer forget some of the things you typed, it also keeps track of how many characters you typed before (and including) RETURN.

Now, if we are in immediate mode, we can make the computer think we typed something in by PRINTing it, then RETURNing over what was printed on screen. The dynamic keyboard routine involves PRINTing a command on the screen and then POKEing a few carriage returns (13) into the buffer to make the computer think *we* typed in the command and the carriage returns. This way we only need to POKE one carriage return for every line we want entered.

There are a few drawbacks to this system. First of all, it only works in immediate mode, not as an executing, RUNning program. So, we have to PRINT the command and PRINT a GOTO to get us back into the program. This requires the cursor to be very carefully positioned each time we execute a command – which means no PRINT statements can be anywhere in our EXEC file. Second, INPUT, INPUT#, GET, and GET# are illegal in immediate mode and therefore cannot be used in our EXEC file. And last, since typing in a line with a line number causes that line to be added to the program, our EXEC file will have no line numbers. This means that any use of GOTOs or GOSUBs will call lines in the *program*, not in the EXEC file. Be very careful if you use these commands.

## The Program

"EXEC-file" was written on a VIC-20 and will also run, as is, on a Commodore 64 (you may want to change the "22" in line 85 to "40"). It can easily be modified to run on other Commodore machines (more on this later).

| Lines | Function |
|---|---|
| 40-60 | get input and store file to disk |
| 70-100 | call and execute the file |
| 1000-1002 | check for disk error |

When creating an EXEC file, be sure to type in the EXEC file commands *without* line numbers. Numbers will almost guarantee a crash when you later EXEC the file. The file-call routine (lines 70-100) can easily be lifted and relocated to be used in another program.

## Modifications

Users of other Commodore machines should find this program very easy to modify for their system. There are only two changes to be made.

First, change the exit code (the key you press to stop creating the file and get on to other things). The exit code is in quotes on line 55; change the prompt in line 40 accordingly.

# HERE'S SOME GREAT GIFT IDEAS FOR YOUR

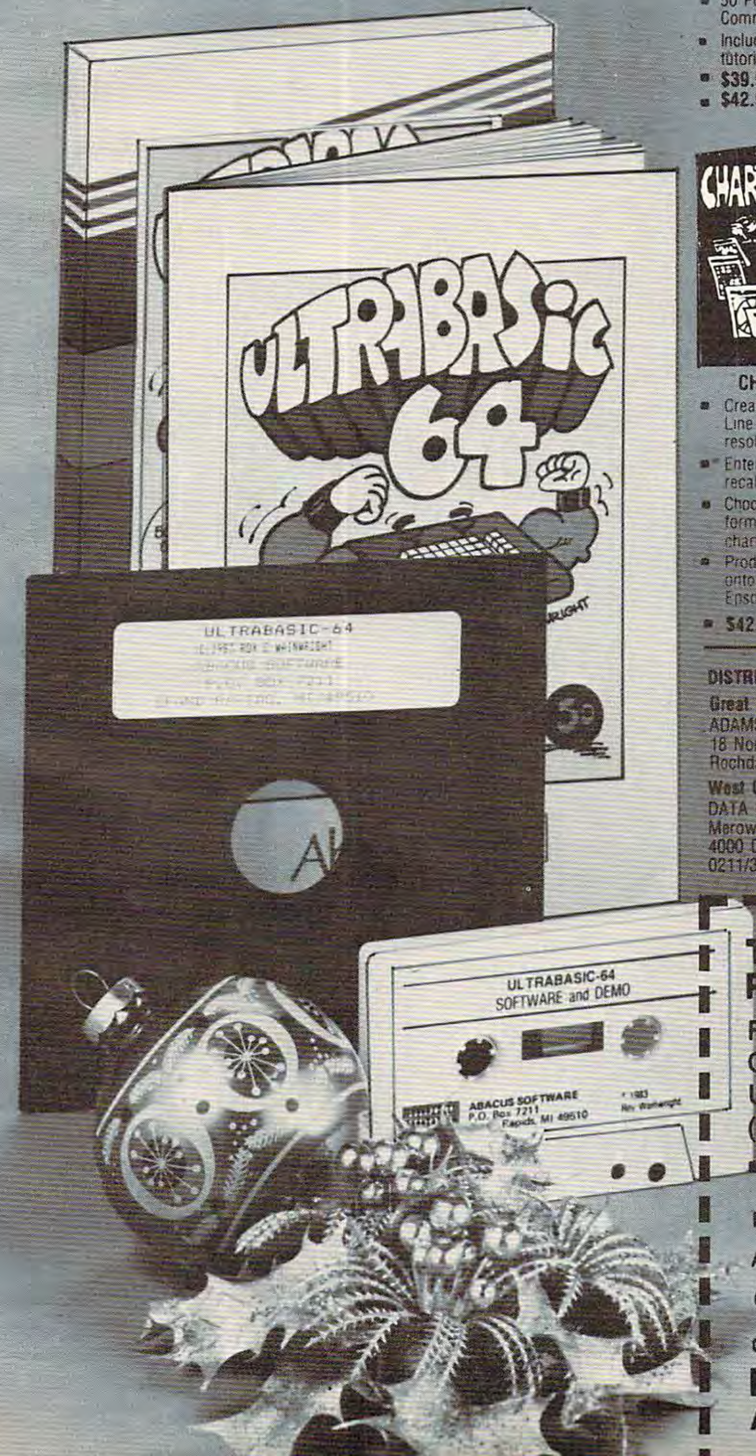
# COMMODORE-64

## OR VIC-20 Computer

**Every ABACUS SOFTWARE Package includes an attractive outer carton, a manual and a disk or cassette.**

YOU CAN COUNT ON

Abacus Software





**ULTRABASIC 64**
- TURTLE, hires, multicolor and sprite graphics.
- Sound and effects.
- Hardcopy.
- 50 Powerful Commands.
- Includes demos, tutorial, manual.
- $39.95 FOR TAPE
- $42.95 FOR DISK


**SUPER DISK UTILITY 64**
- Single Disk Copy (3 types) or Append
- Dump or Modify Any Disk Sector (Has Scrolling)
- Directory Display with scrolling
- BAM Display
- Printer Output.
- Fast Machine Code
- $22.95 DISK ONLY

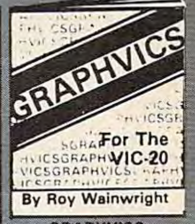

**SYNTHY 64**
- Full featured music and sound synthesizer.
- Easy entry and editing of notes and commands.
- Control ASDR, filters, waveforms etc.
- Includes sample music and manual.
- $29.95 FOR TAPE
- $32.95 FOR DISK


**CHECKBOOK MANAGER 64**
- Menu/Function Key makes it very easy to use.
- Complete editing.
- Full checkbook balancing.
- Printer Output.
- $22.95 DISK ONLY


**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.
- $42.95DISK ONLY


**SKIER 64**
- Arcade type game.
- Joystick or keyboard control.
- 3 different games.
- 3 different levels.
- $14.95 FOR TAPE
- $17.95 FOR DISK


**TINY FORTH**
- Easily extendable language.
- Based on fig-FORTH.
- Commodore 64 or Vic-20 w/8k exp.
- $19.95 FOR TAPE
- $22.95 FOR DISK




**GRAPHVICS**
- Hires and multicolor graphics.
- Fast ploting at machine language speed.
- Two screens-graphics and text.
- Save screen to tape/disk.
- VIC-20 w/3k or 8k exp.
- $19.95 FOR TAPE
- $22.95 FOR DISK

**DISTRIBUTORS**

**Great Britain:** ADAMSOFT, 18 Norwich Ave., Rochdale, Lancs.

**West Germany:** DATA BECKER, Merowingerstr 30, 4000 Dusseldorf, 0211/312085

**Great Britain:** CCI Software, 167 Great Portland St., London WI, 01-636-6354

**Sweden:** TIAL TRADING, PO 516, 34300 Almhult, 476-12304

**Canada East:** KING MICROWARE LTD., 5950 Cote des Neiges, Montreal, Quebec H3S 1Z6, 514/737-9335

**Australia:** CW ELECTRONICS, 416 Logan Road, Brisbane, Queens., 07-397-0808

**Canada West:** L.S.I. Distributors Ltd., 810 W Broadway #163, Vancouver, BC V5Z 4C9, 604/733-0211

**New Zealand:** VISCOUNT ELECTRONICS, 306-308 Church Street, Palmerston North, 63-86-696

**TO ORDER NOW: Write Abacus Software P.O. Box 7211 Grand Rapids, Mich. 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. (Michigan Residents add 4% sales tax).

**FOR QUICK SERVICE PHONE 616/241-5510**

NAME__________

ADDRESS__________

CITY__________ STATE_____ ZIP_____

CHARGE CARD__________ NUMBER__________

**FREE CATALOG** Ask for a listing of other Abacus Software for Commodore-64 or Vic-20

Next, the keyboard buffer and the "how-many" (number of characters currently contained in keyboard buffer) location are in different places on different machines. The chart below should assist you in changing this (in line 90).

| | VIC/64 | Original ROM PETs | BASIC 4.0 PETs |
|---|---|---|---|
| *buffer* | 631-640 | 527-536 | 623-632 |
| *how-many* | 198 | 525 | 158 |

Thus, on a PET 4032, line 90 would become:

```
90 POKE 623,13:POKE624,13:POKE625,13:POKE
   626,13:POKE158,4:STOP
```

VIC and 64 owners may also wish to make the print color the same as the screen color before calling this routine, so the user is unaware of the EXEC taking place on screen.

## Testing The EXEC

The program here contains both the filemaking and EXEC routines. To test the EXEC function, you must first answer YES when asked if you want to "Create A File?" and then type something like A=51:B=17:F$="Mirabelle" or whatever you want to pass to the program from the disk. It could be a POKE to change screen color or to change the character set, anything you like.

Then, the special file will be on your disk under whatever name you gave it during the filemaking phase. To try out the file, RUN the program again, but answer NO when asked if you want to create a file. This time, the program will move down to line 70 and EXEC the file. When you use the EXEC function in a program, you'll probably want to replace F$ in line 70 with the actual name of the file you want to EXEC: OPEN2, 8,2,"NAME,U,R". The technique of *adding* a string to a quoted name (F$+",U,R") is the way to specify variable file names, but in a real program you'll know in advance the file name that you intend to EXEC.

Pay special attention to the key in quotes in line 55. If you are using a PET/CBM, for example, you'll want to change this to the back-arrow key or something. PET/CBM has no function keys so you could never signal the end of your INPUT when creating an EXEC file.

### EXEC-file

```
10 REM *COMMODORE*
15 REM *EXEC-FILE*
20 K$="":A$="":F$=""
25 INPUT"{CLR}{2 DOWN}CREATE A FILE";A$
30 INPUT"{2 DOWN}FILE NAME";F$
35 IFLEFT$(A$,1)<>"Y"THEN70
38 REM**CREATE EXEC FILE **
40 PRINT"{CLR}{3 DOWN}{RVS}{6 SPACES}F1
   TO END{7 SPACES}"
45 OPEN2,8,2,"@0:"+F$+",U,W":OPEN15,8,15
   :GOSUB1000
50 GETA$:IFA$<>""THENPRINTA$;:PRINT#2,A$
   ;
55 IFA$<>"{F1}"THEN50
60 CLOSE2:CLOSE15
66 REM**{3 SPACES}EXECUTE FILE{2 SPACES}
   **
70 OPEN2,8,2,F$+",U,R":OPEN15,8,15:GOSUB
   1000
75 PRINT"{CLR}{4 DOWN}":GET#2,A$:IFA$<>C
   HR$(13)AND(ST<>64)THENK$=K$+A$:GOTO75
80 IF (ST)AND64 THEN 100
85 PRINTK$"{3 DOWN}":PRINT"GOTO75":PRINT
   "{10 UP}"+LEFT$("{7 UP}",INT(LEN(K$)/
   22)):K$=""
90 POKE631,13:POKE632,13:POKE633,13:POKE
   634,13:POKE198,4:STOP
100 CLOSE2:CLOSE15:END
911 REM..{3 SPACES}CHECK DISK ERROR
    {2 SPACES}..
1000 INPUT#15,EN,EM$,ET,ES
1001 IFDS>20THENPRINTEN,EM$:STOP
1002 RETURN
63003 A=PEEK(B)+256*PEEK(B+1):IFA=0THENC
      LR:END
```

©

# Atari Master Disk Directory

Joseph M. Apice

*With this program you create a single disk "library" incorporating the contents of all your directories. The menu gives you six options – the program is truly multipurpose.*

Master directories are an essential part of any computer system. We often take them for granted in the larger minicomputers simply because they exist. These multi-user systems utilize some kind of central library containing a list of all the user directories and their files.

In our smaller home computers, we do not have this luxury. And after working on a mini all day, I find it difficult to do without a master directory so I decided to incorporate some of the nicer features of the larger system into my personal computer. Though it is impossible to exactly duplicate the features, I found I could make a reasonable addition.

I had read several articles dealing with various types of master directory programs. All of them were good, but many required the constant swapping of disks. I needed something that could quickly display the contents of any directory in my library as well as locate any file that I wanted to use without searching through my entire library.

With this in mind, I used the Atari forced read mode to load the contents of every directory in my library as a series of DATA statements in the "Master Disk Directory" program. I could then use the program to examine the contents of any disk, search for any file, and even print labels for my disks without loading any other disk.

The program is menu driven and structured so that each menu function is a subroutine. This allows the user to follow what is being done and to make any desired changes.

## Running The Program

After you load the program and type RUN, a menu will display the six options available. Enter the number preceding your selected option and press RETURN.

1. *Directory Update.* This first option is selected each time you enter a new disk or update the listing of a previous disk into the master directory. At the prompt, simply enter the disk name or label and press RETURN. Any additional files which may have existed in the previous disk are automatically deleted when the most recent copy is installed.

2. *Disk Search.* Use this option to review the contents of any disk directory previously installed. Enter the name of the disk you wish to view, and the contents of that disk directory will be displayed to the screen.

3. *File Search.* One interesting feature of the program is that it can quickly locate any named file and its resident disk. The wild card feature is always active if the full name is not specified. Multiple listings of any file will be displayed along with their disk locations. The message NO MATCH FOUND will be displayed if the named file does not reside on any disk.

4. *Print Labels.* Those of you who own a Gemini 10 or Epson MX-80 compatible printer can use this option to print directory labels. The program will allow up to 24 files and one header on any standard (4 x 1-7/16 inch) label. Additional files are printed on the next label.

5. *Install Update.* When you have completed the transfer of all the directories, use this option to install the most recent update into the Master Disk Directory program. The SAVE feature is automatic; when it is completed, the program will return you to the main menu.

# ComputAbility™

## The Discount Software Company That Pays You A Dividend! !

ATARI

**ATARI 800 48K ........CALL**
ATARI 800XL 64K .................CALL
ATARI 810 DISK DRIVE .................CALL
ATARI 1050 DISK DRIVE.................CALL
ATARI 850 INTERFACE .................CALL
ATARI 1010 RECORDER.................77.00
HOME MANAGER KIT.................64.95
ATARI 1027 PRINTER .................289.00

**MOSAIC**
64K Ram/400.................149.00
64K Ram + Cable Kit /400/800 ......169.00
48K Ram Kit.................94.00
16/32 Expander.................64.95
32K Ram.................77.95
Mosaic Adaptor.................49.95

DISK DRIVES: **PERCOM, RANA, TRAK, ASTRA**..........CALL FOR PRICES.

**Suncom Joysensor...34.95**
30 day money back guarantee

**PRINTERS**
PROWRITER 8510 AP ... CALL
**Seikosha**
**W/Parallel Port .........CALL**

## SUPER SPECIALS

| BLUE MAX | ZORK I | JUMPMAN JR. | WITNESS | RIVER RESCUE | AE |
|---|---|---|---|---|---|
| $22.95 DISK/TAPE | $26.95 DISK | $26.95 CARTRIDGE | $33.95 DISK | $26.95 CARTRIDGE | $22.95 DISK |

PRICES EFFECTIVE NOW THROUGH OCTOBER 31, 1983

## FREE* SOFTWARE AND ACCESSORIES

### ATARI
Conversational Languages - T...43.95
Programming 2 & 3 - T.........21.95
Music Composer - C...........32.95
My First Alphabet - D..........26.95
Touch Typing - T...............19.95
Home Filing Manager - D......37.95
Mailing List - T.................19.95
Asteroids - C..................26.95
Caverns Of Mars - D...........28.95
Computer Chess - C...........26.95
Missile Command - C..........26.95
Super Breakout - C............26.95
Star Raiders - C................32.95
Assembly Editor - C...........44.95
Basic - C.......................39.95
Macroassembler - D...........65.95
Microsoft Basic - D...........65.95
Pilot (Home Package) - C......58.95
Invitation To Programming I - T...18.95
Speed Reading - T.............55.95
Basketball - C..................26.95
Graph-It - T....................15.95
Juggles House - D/T...........22.95
Pilot (Educator) - C.............97.95
Video Easel - C................26.95
Defender - C....................32.95
Galaxian - C....................32.95
Qix - C..........................32.95
Dig Dug - C.....................32.95
ET - C...........................37.95
Timewise - D....................23.95
Atariwriter - C..................74.95
Donkey Kong - C...............37.95
Ms. Pac-Man - C...............39.95
Tennis - C.......................35.95
Eastern Front - C..............32.95
Donkey Kong Jr. - C...........39.95
Pengo - C........................35.95
Logo - C.........................79.95
Robitron - C....................35.95
Pole Position - C...............39.95
Microsoft Basic II - C..........65.95
Paint - D.........................33.95
Caverns of Mars - C............32.95

### FIRST STAR
Astro Chase - D/T..............23.95
Bristles - D/T...................23.95
Flip Flop - D/T..................23.95
Rent Wars - D/T................23.95

### BIG FIVE
Miner 2049'er - Cart...........39.95
Scraper Caper - C.............39.95

### COMPUTABILITY DIVIDEND COUPON PREMIUMS

| | |
|---|---|
| 3 M Blank Disk - Box of 10 | 27.00 or 10 Coupons |
| Elephant Disks - Box of 10 | 22.00 or 8 Coupons |
| Flip 'n File Disk Holder | 20.95 or 6 Coupons |
| Flip 'n File Cartridge Holder | 20.95 or 6 Coupons |
| Monitor Stand | 27.95 or 10 Coupons |
| VU - Case Disk Holder W/Lock | 24.00 or 10 Coupons |
| Heavy Plastic Dust Cover 800 or 810 | 9.95 or 3 Coupons |
| First Book of Atari Software 1983 | 17.95 or 6 Coupons |
| Kids & The Atari | 17.95 or 6 Coupons |
| The Atari Assembler | 12.95 or 5 Coupons |
| Atari Games & Recreation | 14.95 or 6 Coupons |
| Atari Pilot for Beginners | 14.95 or 6 Coupons |
| The Visicalc Book Atari | 14.95 or 6 Coupons |
| Atari Basic | 12.95 or 5 Coupons |
| Atari Sound & Graphics | 9.95 or 3 Coupons |
| Starfighter | 12.99 or 5 Coupons |
| Slik Stik | 9.95 or 3 Coupons |
| Extension Cable - 6 ft. | 6.95 or 3 Coupons |
| Lefty Adaptor | 9.95 or 3 Coupons |
| Stik Stand | 6.99 or 3 Coupons |
| Suncom Tac II | 19.95 or 8 Coupons |
| Suncom Game Switch | 6.95 or 3 Coupons |
| TI Adaptor | 12.95 or 5 Coupons |
| Kraft Switch Hitter Joystick | 17.95 or 7 Coupons |
| Kraft Joystick | 15.95 or 6 Coupons |
| Maxell 2 Pack Disks | 8.95 or 3 Coupons |
| Inside Atari Basic | 12.95 or 5 Coupons |
| Compute's First Book of Atari | 12.95 or 5 Coupons |
| Compute's Second Book of Atari | 12.95 or 5 Coupons |
| De Re Atari | 17.95 or 6 Coupons |
| Wico Joystick | 24.00 or 10 Coupons |
| $24.00 Worth of Software | 10 Coupons |

### *COUPON PROGRAM
The purchase of each program (with the exception of Super Specials, Atari and APX) will earn you 1 COMPUTABILITY DIVIDEND COUPON. Save 10 coupons and redeem them for your choice of any program we sell for $24.00 or less. Less than 10 coupons may be redeemed for premium items as indicated. You pay only a $2.50 shipping & handling charge.

### DATASOFT
Pooyan - D/T...................23.95
Moon Shuttle - D/T............23.95
Sands of Egypt - D............23.95
Text Wizard - D................37.95
Spell Wizard - D...............37.95
Basic Compiler - D............59.95
Teletalk - D....................37.95
Canyon Climber - D/T.........20.95

### SIRIUS
Bandits - D.....................27.95
Way Out - D....................31.95
Repton - D.....................31.95
Twerps - D.....................27.95
Blade of Blackpoole - D........31.95
Type Attack - D................31.95
Beer Run - D...................27.95
Critical Mass - D..............31.95

### ADVENTURE INTERNATIONAL
Preppie - D/T..................23.95
S.A.G.A. Adventures - D........31.95
Sea Dragon - D/T..............27.95
Stratos - D/T...................27.95
Bugoff - D/T....................23.95
Preppie II - D/C................27.95
Stone of Sisyphus - D..........31.95
Eliminator - D/T................20.95
Rally Speedway - Cart..........37.95
Ultra Disassembler - D.........39.95
Diskey - D......................39.95
Triad - D/T......................27.95

### SYNAPSE
Dimension X - D/T.............27.95
Shadow World - D/T............27.95
Rainbow Walker - D/T..........27.95
Drelbs - D/T....................27.95
Shamus II - D/T.................27.95
Blue Max - D/T..................27.95
Fort Apocalypse - D/T..........27.95
Necromancer - D/T.............27.95
Pharoah's Curse - D/T.........27.95
Survivor - D/T..................27.95

### PARKER BROS.
Q-Bert
Frogger
Astro Chase
Popeye
Tutankhamen
Risk
Chess
Please call for Prices

### MISCELLANEOUS
Night Mission - D/T............23.95
Jumpman - D/T.................31.95
Juice - D/T......................23.95
Castle-Wolfenstein - D..........23.95
Financial Wizard - D............44.95
PM Animator - D................27.95
Teletari - D.....................31.95
Starbowl Football - D/T.........25.50
Ali Baba - D....................26.50
Ultima II - D....................44.95
Crisis Mountain - D............27.95
Star League Baseball - D/T.....25.50
Home Accountant - D............54.95
Mountain King - Cart............31.95

**D - Disk  T - Cassette  C - Cartridge**

ATARI is a trademark of ATARI, Inc.

**We Carry Hundreds of Items for ATARI 400/800, Ask for Our FREE CATALOG.**

**Mastercard/VISA Order Toll Free** MasterCard **800-558-0003** VISA **In Wisc. Call 414/351-2007**

No surcharge for credit cards

COMPUTABILITY
P.O. Box 17882
Milwaukee, WI 53217

**ORDERING INFORMATION:** To order by mail, send money order, certified check, or personal check (allow 14 days to clear) to COMPUTABILITY. Include $2.00 shipping on all software orders. Include 3% shipping on all hardware orders, minimum $2.50. Mastercard and Visa please include card number and expiration date. WI residents please add 5% sales tax. Canadian, APO & FPO software orders include 5% shipping minimum $5.00. All other foreign software, please add 15% shipping, minimum $7.00 (US Funds only). Prices subject to change without notice.

ORDER HOURS
Mon-Fri 12-9 PM (CST)
Sat 12-5 PM (CST)

**6.** *Exit.* This option allows you to exit the program and return to BASIC. A word of caution here: This option should be used after option 5 if any updates are being made as it will erase the Master Disk Directory program from memory when it is selected.

## DATA Locations

Each directory group of DATA statements is allowed a maximum of 64 lines. This corresponds to the maximum number of data files allowed by DOS on any one disk.

Line 2000 will be the first DATA line. Do not renumber the program without making the necessary changes to the variables LINE,LINCNT, and FIRST.

### Master Disk Directory

```
120 DIM A$(20),F$(18),R$(1),SP$(2),L
    B$(20),TAB$(8)
130 STP=65:FIRST=2000:TAB$="
    {6 SPACES}"
150 GRAPHICS 0:POKE 710,146:POKE 712
    ,146:POKE 752,1
160 POSITION 9,4:? "MASTER DIRECTORY
     FILE"
190 FOR PAUSE=1 TO 500:NEXT PAUSE
200 ? CHR$(125):POSITION 13,2:? "MEN
    U SELECTION"
210 POSITION 13,5:? "1..DIRECTORY UP
    DATE"
220 POSITION 13,7:? "2..DISK SEARCH"
230 POSITION 13,9:? "3..FILE SEARCH"
240 POSITION 13,11:? "4..PRINT LABEL
    S"
250 POSITION 13,13:? "5..INSTALL UPD
    ATE":?
260 POSITION 13,15:? "6..EXIT"
280 ? :? :? "ENTER CHOICE---->";:INP
    UT CHOICE
290 ON CHOICE GOSUB 310,670,980,1180
    ,1540,1620
300 GOTO 200
310 REM ** DIR. UPDATE ROUTINE **
320 ? CHR$(125):POSITION 2,6:? "INSE
    RT DISK TO CATALOG IN DRIVE 1"
330 POSITION 2,10:? "DISK LABEL--->
    ";:INPUT LB$
340 TRAP 390:LINCNT=FIRST
360 RESTORE LINCNT:READ A$,N
370 IF A$=LB$ THEN 390
380 LINCNT=LINCNT+STP:GOTO 360
390 LINE=LINCNT:TRAP 520
410 FILCNT=1:OPEN #1,6,0,"D:*.*"
420 INPUT #1,A$
430 ? CHR$(125)
440 ? :? LINCNT+FILCNT;" DATA ";A$:G
    OSUB 470
450 FILCNT=FILCNT+1:GOTO 420
470 ? :? :? "CONT"
480 POSITION 0,0
490 POKE 842,13:STOP
500 POKE 842,12
510 RETURN
520 TRAP 40000:CLOSE #1
530 ? CHR$(125):? :? LINE;" DATA ";L
    B$;",";FILCNT-1
540 GOSUB 470
550 IF FILCNT>=N THEN 630
560 DLINE=LINCNT+FILCNT
570 FOR I=FILCNT-1 TO N
580 ? CHR$(125)
590 ? :? DLINE
600 GOSUB 470
610 DLINE=DLINE+1
620 NEXT I
630 ? CHR$(125):POSITION 8,6:? "ANY
    MORE DISKS";:INPUT A$
640 IF A$="Y" OR A$="YES" THEN 320
650 POSITION 6,12:? "REMOVE DISK PRE
    SS--->RETURN";:INPUT A$
660 LINCNT=FIRST:RETURN
670 REM ** DISK SEARCH ROUTINE **
680 ? CHR$(125):POSITION 2,2:? "SEAR
    CH WHICH DISK-->";:INPUT LB$
690 ? CHR$(125)
700 SP$="  ":LINCNT=FIRST:TRAP 950
730 RESTORE LINCNT:READ A$,N
740 IF A$=LB$ THEN 760
750 LINCNT=LINCNT+STP:GOTO 730
760 L=LEN(A$):CENT=20-INT(L/2)
770 POSITION CENT,0:? A$:POSITION CE
    NT,1
790 FOR I=1 TO L:? "-";:NEXT I
800 POSITION 13,4:? "DISK DIRECTORY"
    :?
810 FOR I=1 TO N
820 IF I>=10 THEN SP$=" "
830 IF I=17 OR I=34 OR I=51 THEN 930
840 READ A$
850 IF A$(4,5)=" F" THEN 910
860 IF A$(1,2)<>"* " THEN 890
870 PRINT TAB$;SP$;I;" ";A$
880 GOTO 900
890 PRINT TAB$;SP$;I;"{3 SPACES}";A$
900 NEXT I
910 ? :? TAB$;"{3 SPACES}";A$:GOTO 9
    40
930 ? :? "PRESS---> RETURN TO CONTIN
    UE";:INPUT R$:? "{CLEAR}":GOTO 8
    40
940 ? :? :? "PRESS---> RETURN TO CON
    TINUE";:INPUT R$:RETURN
950 TRAP 40000
960 ? CHR$(125):? :? :? "DISK ---> "
    ;LB$;"{5 SPACES}NOT FOUND"
970 ? :? "PRESS RETURN TO CONTINUE--
    >";:INPUT A$:RETURN
980 REM ** FILE SEARCH ROUTINE **
990 ? CHR$(125):POSITION 2,2:? "SEAR
    CH WHICH FILE-->";:INPUT F$
1000 LINCNT=2000:? CHR$(125)
1010 RESTORE LINCNT:TRAP 1150
1020 READ A$,N
1030 PRINT "SEARCHING DISK---> ";A$:
     ?
1040 FOR I=1 TO N
1050 READ A$
1060 IF A$(1,2)<>"* " THEN 1090
1070 IF A$(3,LEN(F$)+2)=F$ THEN PRIN
     T "FILE LOCATED---> ";A$:? :FLA
     G=1
1080 GOTO 1100
1090 IF A$(1,LEN(F$))=F$ THEN PRINT
     "FILE LOCATED---> ";A$:? :FLAG=
     1
1100 NEXT I
1120 LINCNT=LINCNT+STP:GOTO 1010
1140 GOTO 1020
1150 IF FLAG THEN 1170
1160 ? CHR$(125):POSITION 8,16:? "--
     NO MATCH FOUND--"
```

```
1170 FLAG=0:? :? "LIST EXHAUSTED ---
     ->RETURN";:INPUT R$:RETURN
1180 REM ** DISK LABEL ROUTINE **
1190 ? CHR$(125):POSITION 8,4:? "LOA
     D PRINTER WITH LABELS"
1200 POSITION 7,6:? "PUT 850 INTERFA
     CE-ON LINE-"
1210 POSITION 10,7:? "PUT PRINTER-ON
      LINE-"
1220 POSITION 8,10:? "PRESS RETURN W
     HEN READY";:INPUT A$
1230 TRAP 1520:ROW=0:COL=0:INC=12
1240 CLOSE #2:OPEN #2,8,0,"P:"
1250 LPRINT CHR$(27);"@":GOTO 1280
1260 PUT #2,27:PUT #2,51:PUT #2,18
1270 ? #2;CHR$(15);:? #2;CHR$(27);CH
     R$(83);:RETURN
1280 ? CHR$(125):POSITION 2,4:? "PRI
     NT LABELS FOR WHICH DISK ":? :?
      "---> ";:INPUT LB$
1300 LINE=FIRST:TRAP 1530
1310 RESTORE LINE:READ A$,N
1320 IF A$=LB$ THEN 1340
1330 LINE=LINE+STP:GOTO 1310
1340 PUT #2,27:PUT #2,71
1350 ? #2;" DISK= ";A$
1360 PUT #2,27:PUT #2,64
1370 GOSUB 1260:PRINT #2
1380 ROW=ROW+3
1390 FOR I=1 TO N
1400 READ A$
1410 IF A$(4,5)=" F" THEN 1480
1420 IF A$(1,2)<>"* " THEN 1450
1430 ? #2;A$;"  ";:COL=COL+1:ROW=ROW
     +1:IF COL=3 THEN PRINT #2:COL=0
1440 GOTO 1460
1450 ? #2;"  ";A$;"  ";:COL=COL+1:RO
     W=ROW+1:IF COL=3 THEN PRINT #2:
     COL=0
1460 IF ROW=24 THEN ROW=0:PRINT #2:P
     RINT #2
1470 NEXT I
1480 PRINT #2;"  ";A$
1490 SKIP=INT(ROW/3)
1500 FOR I=1 TO INC-SKIP:PRINT #2:NE
     XT I
1510 RETURN
1520 TRAP 40000:GOTO 1190
1530 TRAP 40000:GOTO 1280
1540 REM ** EXIT TO BASIC & SAVE UPD
     ATED PROGRAM **
1550 ? CHR$(125):POSITION 3,8:? "INS
     ERT DISK CONTAINING MASTER.DIR"
     :POSITION 14,10:? "IN DRIVE #1"
     :POSITION 3,13
1560 ? "PRESS ---> RETURN{5 SPACES}W
     HEN READY";:INPUT A$
1570 ? CHR$(125):? :? "SAVE ";CHR$(3
     4);"D:MASTER.DIR"
1580 GOSUB 470
1590 ? "{CLEAR}":POSITION 12,4:? "UP
     DATE INSTALLED"
1600 FOR PAUSE=1 TO 500:NEXT PAUSE
1610 RETURN
1620 GRAPHICS 0:NEW
```

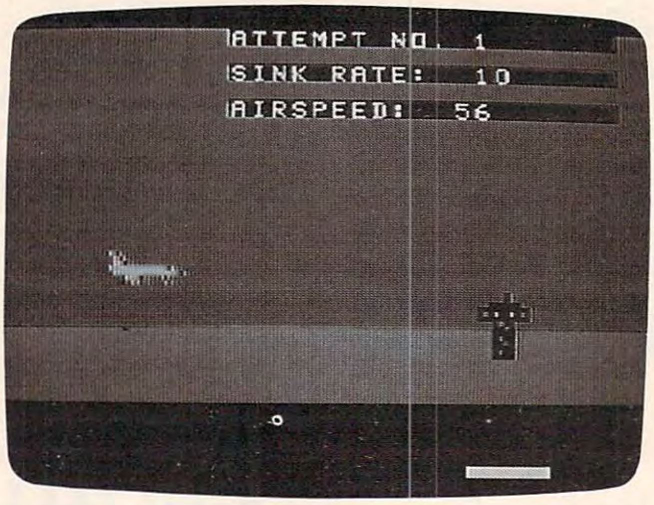
©

# Runway 180

## Using Sprites In TI Extended BASIC

James Dunn

*The efficient, remarkable sprite-handling ability of TI Extended BASIC is clearly evident in this game. The author discusses creating sprites and explores sprite manipulation. There are several valuable pointers here for those interested in graphics, animation, or game programming on the TI.*




*Your plane is on final approach. "Runway 180," TI version.*

### Using Sprites In TI Extended BASIC

One of the biggest problems in designing an arcade-type game in BASIC is that BASIC can move only one character at a time, usually slowly and usually not very smoothly. Ideally, we need the ability to move an object independently of the operation of the main program. Once set in motion, the object would continue in motion until acted upon by a new command from the main program. Sprites accomplish this.

Although a sprite is a type of subprogram that runs concurrently with a main program, the main program first must create the sprite, define its shape, and set it in motion. A sprite then continues its motion without requiring continuous control from the main program, except that the main program may at any time test the sprite for position, change the color or pattern, delete, or change its motion.

Included in TI-99/4A Extended BASIC are 11 commands to control sprites: CALL COLOR, CALL CHAR, CALL SPRITE, CALL PATTERN, CALL MAGNIFY, CALL MOTION, CALL POSITION, CALL LOCATE, CALL DISTANCE, CALL COINC, and CALL DELSPRITE. To illustrate the use of these commands, we'll look at an airplane landing game, "Runway 180." Try some examples for yourself to get a feel for sprite programming.

### Creating Sprites

Certain considerations must be taken into account before sprites are created. If a special graphics character is to be used for the sprite, the character must be created by use of CALL CHAR. For example, in the game there are three special characters defined for the aircraft. One is with the wheels up (lines 430)-460), one is with the wheels down (lines 510-540), and one is debris after a crash (lines 550-580).

To create a special character, it is necessary to redefine an existing standard character. The standard characters correspond to the numbers 30 through 143 (part of what's called the ASCII number code). The new pattern is created by using CALL CHAR and is referenced by its ASCII number.

Before we choose which ASCII number to use, we must examine some other factors. CALL MAGNIFY can enlarge a sprite to one of four magnification factors. Factor four is used in the game (line 630). This enlarges the sprites to double-size pixels and uses a block of four sequential characters. The ASCII number used to define the sprite must be evenly divisible by four and represents the upper-left character in the block of four. The next three ASCII numbers represent the lower-left, upper-right, and lower-right characters respectively in the block of four.

The sprite may be colored independently of the other characters in the same character set. In addition, the sprite with the lower sprite number (this is a different number than the ASCII number) will pass in front of (that is, *over*) the higher numbered sprite. Since the aircraft should pass in front of the tower, it should have a lower sprite number for each of its three configurations (line 610).

To set up a list of sprites, first number the lines on a sheet of paper from 30 to 143. Then, beside each number, write what set it belongs to (set 0 to 14). Since you may want to use letters or numbers in a screen display at the same time, mark out ASCII numbers 48 through 57 and 65 through 90. The remaining ASCII numbers can be used to define special characters for graphics and sprites.

For sprites, using CALL MAGNIFY (4), select four sequential numbers starting at one of the numbers evenly divided by four. Now you are ready to use CALL SPRITE.

CALL CLEAR will not remove a sprite from the screen. To completely clear the screen, you must also use CALL DELSPRITE (line 1350).

## Sprites In Motion

Now that the sprite has been created, there are two ways of moving it around the screen. Let's call these two methods *absolute* and *relative*. The absolute method uses exact row and column positions via the CALL LOCATE command. The relative method uses row and column motion values via the CALL MOTION command.

The absolute method uses a loop with CALL JOYST to increment row and column variables, and then a CALL LOCATE to move the sprite one step each time the loop is executed. This is analogous to nonsprite methods of animation. The drawback in using this method is that the sprite does not move independently; the main program causes the move. A modified form of this method is used for the stall subroutine (line 1470) and the new approach routine (line 1380).

The relative method is similar, using a loop with CALL JOYST to increment row and column *motion* variables which are used in a CALL MOTION command. This allows the sprites to continue moving independently of the main program. By this method, the runway stripe is moved horizontally only (line 680) and the aircraft vertically only (also line 680).

The sprite's shape may be changed anytime during the program by using CALL PATTERN to substitute a different ASCII character number and therefore a different pattern. When the fire button is depressed (line 1130), the aircraft landing gear comes down (line 1190). The pattern is changed again if the aircraft crashes (line 1720).

## Testing For Game Conditions

During the operation of the program, it may become necessary to test for certain conditions. For example, we see if the aircraft has touched down on the runway (line 690), if the tower has reached the left side of the screen (line 700), or if the aircraft is going off the top of the screen (line 710). CALL COINC is used to test for these conditions.

However, there is a problem with this method. Since the main program tests for coincidence only when CALL COINC is executed and since the sprite moves independently of the main program, it is quite possible to miss an exact coincidence when it occurs. For this reason a tolerance factor is included in CALL COINC. So the test is really for a range of + or - tolerance. If the tolerance is too large, coincidence can be returned too early. If the tolerance is too small, coincidence can be missed altogether. How large the tolerance should be depends upon two things: the speed of the sprite and the speed of the loop which is testing for coincidence.

The test for the tower reaching the left side of the screen is in both the main loop (line 700) and the stall loop (line 1480). The tolerance in the stall loop is much smaller because the execution speed is so fast and the sprite moves so slowly that coincidence is actually read twice before the sprite leaves the tolerance range. Trial and error is the only way to find out how large the tolerance should be.

However, after programming this game, it is obvious that very fast-moving sprites will require tolerance ranges that will make arcade-style, fast-action games nearly impossible in Extended BASIC. The problem is that the coincidence test is executed from the main program. If it were part of the sprite subprogram instead, it would be possible to keep the tolerance very small.

CALL POSITION and CALL DISTANCE both suffer from the same problem as CALL COINC. By the time a position or distance can be computed and returned to the main program, the sprite has moved elsewhere. But it is possible to stop the sprite by using a CALL MOTION before using CALL POSITION or CALL DISTANCE (line 1330), then to restart whatever motion is required.

Despite a few shortcomings, the sprite capabilities in Extended BASIC are remarkable. For true arcade-type play, machine language is still necessary, but Extended BASIC sprites will carry the programmer a lot closer to this goal.

**Runway 180**

```
130 CALL CLEAR :: CALL SCREEN(5)::
    CALL COLOR(1,16,1,2,16,1,3,16,1
    ,4,16,1,5,16,1,6,16,1,7,16,1,8,
    16,1)
140 DISPLAY AT(10,9):USING "RUNWAY
    180"
150 FOR B=0 TO 30 STEP 2 :: CALL SO
    UND(-10,110,30,110,30,2500,30,-
    8,B):: CALL SOUND(-10,110,30,11
    0,30,4000,30,-8,B):: NEXT B
160 CALL CLEAR :: DISPLAY AT(10,9):
    USING "PRESS" :: DISPLAY AT(12,
    9):USING "I-FOR INSTRUCTIONS"
170 DISPLAY AT(14,14):USING "OR" ::
     DISPLAY AT(16,9):USING "G-FOR
    GAME"
180 CALL KEY(0,K,S):: IF S<>1 THEN
    180
190 IF K=71 THEN 330
200 IF K=73 THEN 220
210 PRINT "ALPHA LOCK MUST BE ON" :
    : PRINT :: PRINT "TRY AGAIN" ::
     FOR DELAY=1 TO 200 :: NEXT DEL
    AY :: GOTO 160
220 CALL CLEAR :: PRINT "YOU ARE PI
    LOTING A JET" :: PRINT :: PRINT
     "AIRCRAFT WHICH HAS BEEN " ::
    PRINT :: PRINT "CLEARED TO LAND
     ON": :
230 PRINT "RUNWAY 180." :: PRINT ::
     PRINT :: GOSUB 310
240 CALL CLEAR :: PRINT "USE YOUR J
    OYSTICK TO CONTROL" :: PRINT ::
     PRINT "SINK RATE AND AIRSPEED.
    ": :
243 PRINT "JOYSTICK CONTROL-" :: PR
    INT
245 PRINT "LEFT: ACCELERATE" :: PRI
    NT "RIGHT: BRAKE" :: PRINT "UP:
     DECREASE SINK RATE"
247 PRINT "DOWN: INCREASE SINK RATE
    " :: PRINT
250 PRINT "FIREBUTTON CONTROLS LAND
    ING" :: PRINT :: PRINT "GEAR."
    :: PRINT :: PRINT :: GOSUB 310
    :: CALL CLEAR
260 PRINT "TO RECOVER FROM A STALL"
     :: PRINT :: PRINT "INCREASE AI
    RSPEED ABOVE 60." :: PRINT ::
    PRINT "IF YOU CANNOT STOP BEFO
    RE": :
270 PRINT "TOWER REACHES LEFT SIDE
    OF" :: PRINT :: PRINT "SCREEN,
    INCREASE AIRSPEED" :: PRINT
280 PRINT "TO 60 AND LIFT OFF FOR "
     :: PRINT :: PRINT "ANOTHER PAS
    S." :: PRINT :: PRINT :: GOSUB
    310 :: CALL CLEAR
290 PRINT "YOU MAY HAVE FOUR PASSES
    " :: PRINT :: PRINT "AT THE RUN
    WAY....." :: PRINT :: PRINT "BE
    WARE OF THE WIND SHIFTS!" :: PR
    INT :: PRINT
300 PRINT "GOOD LUCK!!!!" :: PRINT
    :: PRINT :: PRINT :: PRINT :: G
    OSUB 310 :: GO TO 330
310 PRINT :: DISPLAY AT(24,1):USING
     "HIT ANY KEY TO CONTINUE"
320 CALL KEY(0,R8,S8):: IF S8<>1 TH
    EN 320 ELSE RETURN
330 A1=1
340 REM INITIALIZE
350 A=0 :: B=-75 :: LG=0 :: CALL SC
    REEN(2)
360 CALL CLEAR :: CALL CHAR(33,"FFF
    FFFFFFFFFFFFF"):: CALL COLOR(1,
    8,1)
370 LC=0 :: FOR Z=1 TO 16 :: CALL H
    CHAR(Z,1,33,32):: NEXT Z
380 CALL CHAR(42,"FFFFFFFFFFFFFFFF"
    ):: CALL COLOR(2,13,1)
390 FOR Z=17 TO 20 :: CALL HCHAR(Z,
    1,42,32):: NEXT Z
400 RANDOMIZE
410 REM DEF CHAR
420 CALL CHAR(96,"00000000FFFFFFFF
    FFFFFFF0000000000000000FFFFFFFF
    FFFFFFFF")
430 CALL CHAR(120,"0030181C3F1F0700
    ")
440 CALL CHAR(121,"000000")
450 CALL CHAR(122,"00000000FCFF8000
    ")
460 CALL CHAR(123,"00000000")
470 CALL CHAR(104,"0000000071F151F
    ")
480 CALL CHAR(105,"0203030203030203
    ")
490 CALL CHAR(106,"00008080E0F8A8F8
    ")
500 CALL CHAR(107,"C040C0C040C0C0C0
    ")
510 CALL CHAR(124,"0030181C3F1F0705
    0000")
520 CALL CHAR(126,"00000000FCFF8884
    0000")
530 CALL CHAR(125,"00000000")
540 CALL CHAR(127,"00000000")
550 CALL CHAR(128,"000000000021F3B00
    ")
560 CALL CHAR(129,"00000000E56E300
    ")
570 CALL CHAR(130,"00000000")
580 CALL CHAR(131,"00000000")
590 REM DRAW DISPLAY
600 CALL SPRITE(#1,96,2,180,1,0,B):
    : CALL COLOR(#1,16)
610 CALL SPRITE(#2,120,2,10,245,A,0
    ):: CALL COLOR(#2,7)
620 CALL SPRITE(#3,104,2,110,250,0,
    -2)
630 CALL MAGNIFY(4)
640 FOR C5=1 TO 40 :: CALL LOCATE(#
    2,10,C5):: NEXT C5 :: GOSUB 870
650 REM MAIN LOOP
660 GOSUB 1120 :: GOSUB 890
670 IF J=0 THEN 690
680 CALL MOTION(#1,0,B,#2,A,0)
690 CALL COINC(#2,170,40,9,T)
700 CALL COINC(#3,110,1,4,DA)
710 CALL COINC(#2,240,40,9,E):: IF
    E=-1 THEN A=1 :: GOSUB 890 :: G
    OTO 680
720 IF DA=-1 THEN 1320
730 IF T<>-1 THEN 660
740 CALL MOTION(#2,0,0)
750 IF A>1 THEN GOSUB 920 :: GOSUB
```

```
     96Ø :: GOTO 166Ø
76Ø IF LG=Ø THEN 166Ø
77Ø GOTO 176Ø
78Ø REM UPDATE DISPLAY
79Ø IMAGE SINK RATE: ###
8ØØ IMAGE RUNWAY ENDS ### YDS
81Ø IMAGE AIRSPEED: ###
82Ø IMAGE TOUCH DOWN
83Ø IMAGE SINK RATE TOO HIGH
84Ø IMAGE AIRSPEED TOO HIGH
85Ø IMAGE CRASH LANDING
86Ø IMAGE STALL WARNING!
87Ø DISPLAY AT(1,1Ø)SIZE(2Ø):USING
     "ATTEMPT NO. #":A1
88Ø RETURN
89Ø DISPLAY AT(3,1Ø)SIZE(2Ø):USING
     79Ø:A
9ØØ DISPLAY AT(5,1Ø)SIZE(2Ø):USING
     81Ø:-B
91Ø RETURN
92Ø DISPLAY AT(7,5)SIZE(2Ø):USING 8
     3Ø
93Ø RETURN
94Ø DISPLAY AT(7,5)SIZE(2Ø)BEEP:USI
     NG 84Ø
95Ø DISPLAY AT(9,5)SIZE(2Ø):USING "
     BOUNCE" :: RETURN
96Ø DISPLAY AT(9,5)SIZE(2Ø):USING 8
     5Ø
97Ø RETURN
98Ø CALL HCHAR(7,5,33,27):: DISPLAY
      AT(9,5)SIZE(2Ø):USING 82Ø
99Ø RETURN
1ØØØ DISPLAY AT(9,5)SIZE(2Ø):USING
      "WARNING "
1Ø1Ø DISPLAY AT(11,5)SIZE(2Ø):USING
      8ØØ:RE
1Ø2Ø RETURN
1Ø3Ø CALL HCHAR(7,5,33,27):: RETURN
1Ø4Ø CALL HCHAR(9,5,33,27):: RETURN
1Ø5Ø CALL HCHAR(11,5,33,27):: RETUR
      N
1Ø6Ø DISPLAY AT(9,5)SIZE(2Ø):USING
      "LIFT OFF" :: CALL HCHAR(11,5,
      33,27):: RETURN
1Ø7Ø DISPLAY AT(3,1Ø):USING "END OF
       RUNWAY " :: DISPLAY AT(5,1Ø):
      USING "NEW APPROACH" :: DISPLA
      Y AT(7,1Ø):USING "NECESSARY"
1Ø8Ø RETURN
1Ø9Ø PRINT "THAT'S 5 PASSES AT THE"
       :: PRINT :: PRINT "RUNWAY. TU
      RN IN YOUR" :: PRINT :: PRINT
      "PILOT LICENSE AND PUT": :
11ØØ PRINT "SOMEONE ELSE IN THE" ::
       PRINT :: PRINT "COCKPIT" :: P
      RINT :: RETURN
111Ø DISPLAY AT(7,9)BEEP SIZE(2Ø):U
      SING 86Ø :: RETURN
112Ø REM JOYST/ LANDING GEAR
113Ø CALL KEY(1,RV,ST):: IF RV=18 A
      ND LG=Ø THEN 119Ø
114Ø CALL JOYST(1,X,Y):: IF X=Ø AND
       Y=Ø THEN GOSUB 121Ø :: RETURN
115Ø A=A-Y/4 :: B=B+X/4
116Ø IF ABS(A)>127 THEN A=127*SGN(A
      )
117Ø IF B>-5Ø THEN 143Ø
118Ø J=1 :: RETURN
119Ø CALL PATTERN(#2,124)
12ØØ A=A+3 :: B=B+2Ø :: LG=1 :: GOT
      O 116Ø
121Ø REM COMPLICATIONS
122Ø CP=INT(RND*16)
123Ø IF CP=1 THEN B=B-1 :: GOTO 13Ø
      Ø
124Ø IF CP=6 THEN B=B+1 :: GOTO 13Ø
      Ø
125Ø IF CP=1Ø THEN A=A-1 :: GOTO 12
      8Ø
126Ø IF CP=15 THEN A=A+1 :: GOTO 12
      8Ø
127Ø J=Ø :: RETURN
128Ø IF ABS(A)>127 THEN A=127*SGN(A
      )
129Ø GOTO 131Ø
13ØØ IF B<-127 THEN B=-127
131Ø J=1 :: RETURN
132Ø REM NEW APPROACH
133Ø CALL MOTION(#2,Ø,Ø):: CALL POS
      ITION(#2,R4,C4)
134Ø IF A1>4 THEN 14ØØ
135Ø CALL DELSPRITE(#1,#3):: CALL C
      LEAR
136Ø GOSUB 1Ø7Ø
137Ø CALL PATTERN(#2,12Ø)
138Ø FOR X=C4 TO 255 :: CALL LOCATE
      (#2,INT(R4),X):: R4=R4-(R4/(25
      5-C4)):: NEXT X
139Ø A1=A1+1 :: GOTO 34Ø
14ØØ CALL DELSPRITE(ALL):: CALL CLE
      AR
141Ø GOSUB 1Ø9Ø
142Ø FOR DELAY=1 TO 9ØØ :: NEXT DEL
      AY :: GOTO 197Ø
143Ø REM STALL
144Ø GOSUB 111Ø
145Ø CALL MOTION(#2,Ø,Ø)
146Ø CALL POSITION(#2,SR,SC)
147Ø CALL LOCATE(#2,SR,SC)
148Ø CALL COINC(#2,17Ø,4Ø,2,T)
149Ø CALL COINC(#3,11Ø,1,2,DE):: IF
       DE=-1 THEN A1=A1+1 :: GOSUB 8
      7Ø :: IF A1>4 THEN 14ØØ
15ØØ IF T=-1 THEN 166Ø
151Ø SR=SR+4
152Ø CALL KEY(1,RV,ST)
153Ø IF RV=18 AND LG=1 THEN 161Ø
154Ø CALL JOYST(1,X,Y):: IF X=Ø AND
       Y=Ø THEN 147Ø
155Ø B=B+X/4
156Ø REM
157Ø IF B<-6Ø THEN 164Ø
158Ø CALL MOTION(#1,Ø,B)
159Ø GOSUB 89Ø
16ØØ GOTO 147Ø
161Ø CALL PATTERN(#2,12Ø)
162Ø A=A-3 :: B=B-22 :: LG=Ø
163Ø GOTO 156Ø
164Ø GOSUB 1Ø3Ø
165Ø RETURN
166Ø REM CRASH
167Ø CALL MOTION(#1,Ø,Ø,#2,Ø,Ø,#3,Ø
      ,Ø,#4,Ø,Ø)
168Ø CALL SOUND(1ØØØ,-7,Ø)
169Ø FOR P=1 TO 1Ø
17ØØ CALL SCREEN(2)
```

```
1710 CALL SCREEN(16):: NEXT P :: CA
     LL SCREEN(2)
1720 CALL PATTERN(#2,128)
1730 FOR DELAY=1 TO 400 :: NEXT DEL
     AY
1740 CALL DELSPRITE(ALL)
1750 GOTO 1970
1760 REM TOUCHDOWN/BRAKE/T&G
1770 GOSUB 980 :: IF B<-53 THEN 194
     0
1780 CALL JOYST(1,X,Y):: B=B+X/2
1790 IF B>-1 THEN 1880
1800 CALL MOTION(#1,0,B)
1810 CALL COINC(#3,110,1,4,DA)
1820 IF DA=-1 THEN RE=0 :: GOSUB 10
     10 :: GOTO 1660
1830 CALL DISTANCE(#3,110,1,RQ)
1840 RE=INT(SQR(RQ)):: GOSUB 1000 :
     : GOSUB 900
1850 CALL KEY(1,RV,ST):: IF RV=18 A
     ND B<-60 THEN GOSUB 1060 :: A=
     A-2 :: GOTO 187
0
1860 GOTO 1780
1870 CALL MOTION(#2,A,0):: FOR DELA
     Y=1 TO 200 :: NEXT DELAY :: GO
     TO 650
1880 REM SCORING
1890 CALL MOTION(#1,0,0,#2,0,0,#3,0
     ,0,#4,0,0)
1900 CALL DELSPRITE(ALL):: CALL CLE
     AR
1910 PRINT "CONGRATULATIONS !": :
1920 PRINT "YOUR SCORE IS :";(RE/A1
     )*10: :
1930 GOTO 1990
1940 A=A-2 :: CALL MOTION(#2,A,0)::
      GOSUB 940
1950 FOR DELAY=1 TO 20 :: NEXT DELA
     Y
1960 A=A+2 :: GOSUB 1030 :: GOSUB 1
     040 :: GOTO 650
1970 REM PLAY AGAIN
1980 CALL CLEAR
1990 PRINT "PLAY AGAIN (Y/N)?"
2000 CALL KEY(2,RV,SV)
2010 IF SV=0 THEN 2000
2020 IF RV=15 THEN 2050
2030 IF RV=18 THEN 330
2040 GOTO 1990
2050 END
```

©

# How To Create A Data Filing System

## Part 4: The Main Program

Jim Fowler

*In the final installment of this series, the author looks at ways to approach the overall logic of a final system. Safeguards and auxiliary programs are also discussed.*

Now you have most of the detail work on your data file system finished. You know what kind of files you want and how they are formatted. The nature of the output functions (and searches) has determined the coding and index files needed, and this has pretty much dictated the input part of the program. Now we're ready to put it all together.

### Make It Modular

You probably already know the advantages of writing programs with lots of subroutines, each doing a single task. In data filing systems this advantage is particularly obvious. A subroutine to input a string from a particular device is much more useful than one to input a string from the keyboard and another to read a string from the disk. That subroutine can be used in other subroutines, to input an author's name, and again to input the title of a work, the date of publication, and so on. The subroutine to input an author's name and encode a part for the index file can also be used to input a key used to search for a particular author. This can go on and on. Whenever possible, make subroutines so all-purpose that they can be called throughout the program.

In the accompanying flowchart, I have illustrated the design for my author-subject file of books and articles. How you want your data displayed, what you want printed, and what you want on the screen will depend on your individual situation. Some people want a printout of their input as well. It is also easier for some people to proofread text on paper than on a screen, so customize it for your needs. Your requirements will differ from mine, so your flowchart will be different, too. However, you probably should use subroutines in a modular fashion.

### Preventing Disaster

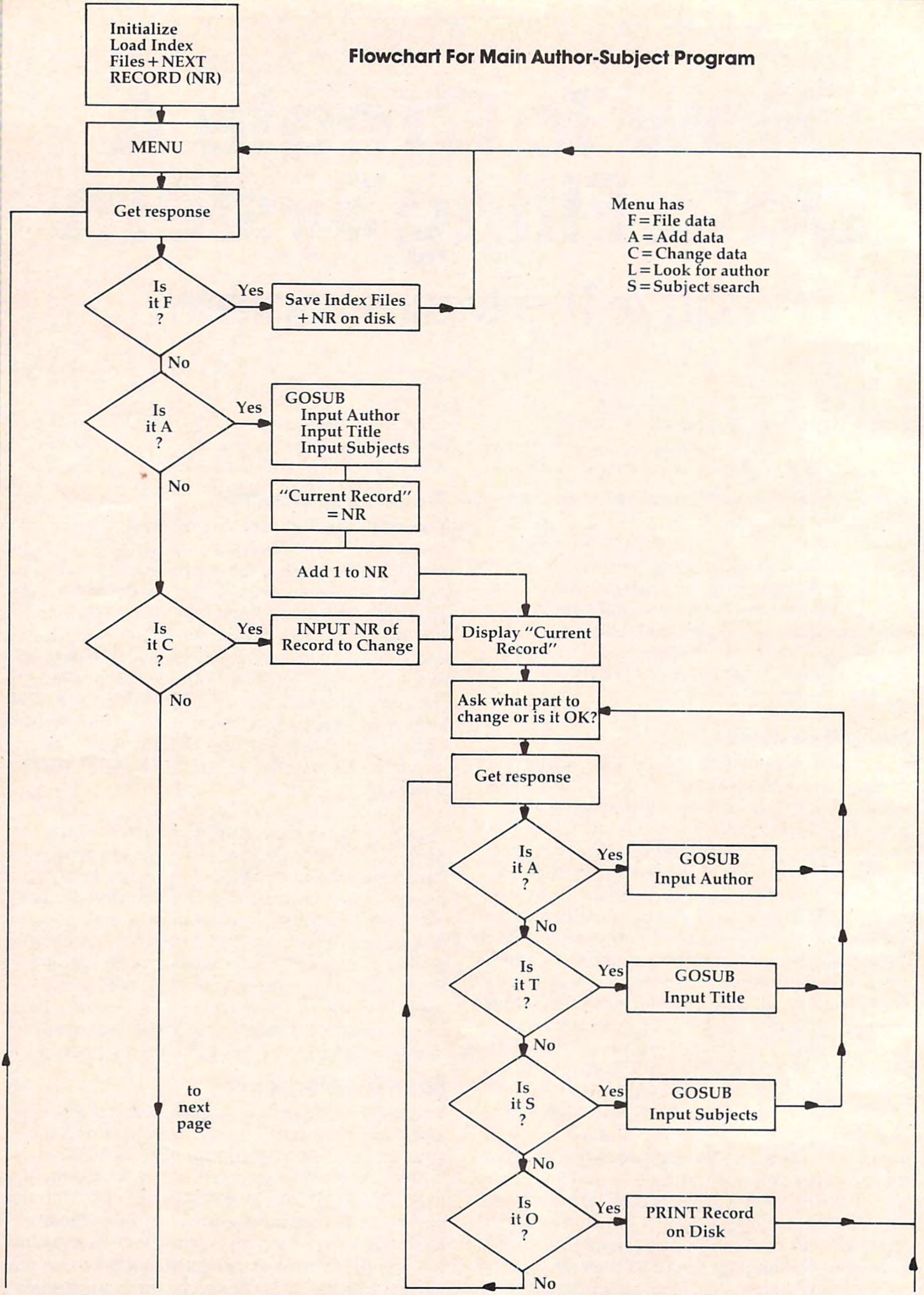
You should include fail-safe methods to prevent disastrous errors. For instance, suppose you have just finished entering a hundred records and you turn off the system without saving the index files. This disaster breaks down into two problems: reminding the user to save the file before quitting, and reconstructing the lost files from the data on disk in the main records. Both are easy to solve, but you must solve them – preferably in advance. Even if your method of reconstructing files is crude or your warning to the user lacks elegance, the important thing is to have these provisions in the program.

You cannot prevent certain disasters, although you can reduce the seriousness of the damage. These include a power interruption, hardware failure, or a bad spot on a disk. To minimize the damage from these troubles, you need good operating procedure. For data files, this means making backup copies frequently. You could, for example, make backup copies after every twentieth entry into the file, then put a counter in the program. When it "goes off," have the program tell you to insert a disk into drive X and "press return" – and there is your backup.

### Satellite Programs

You may find, as I did, that you will need one or more other programs to augment your data file system. For example, you probably will have to write a program to prepare the disk for the records to be written. It should allocate (and fill with nulls) perhaps one thousand relative records. These nulls (zeros) are then replaced as real data is written into the system. A program that does this is a *satellite program*. It is not part of the main system

**Flowchart For Main Author-Subject Program**



# Olympic Sales Company

SERVING YOU SINCE 1947

Telex: 67 34 77 Toll-Free Phone Orders:
Toll-free *(in CA)* 800-252-2153 800-421-8045 (out of CA)
Order Desks open 6 days a week! 7:00 AM to 6:00 PM Mon-Sat
P.O. Box 74545 216 So. Oxford Ave. Los Angeles, CA 90004
Phone: (213) 739-1130 Cable: "OLYRAV" LSA

VISA master charge *We carry close to $5,000,000 inventory at all times. Corp. accts. invited.*

*Good subject to availability; this ad supercedes all previous ads; fob our warehouse; prices subject to change without notice; not responsible for typographical errors; all orders subject to verification; minimum shpg & hndlg $5.95. Send $2 (or $5 foreign) for our famous catalog.*

## COMMODORE 64 $189.95*

COMMODORE 64 Computer—only $189.95*
*when purchased with any of these three packages.

| | | | |
|---|---|---|---|
| COMMODORE 64 | | | $189.95 |
| with the purchase of | | | |
| 1 | 1541 | Disk Drive | $299.95 |
| 1 | 1525E | Printer | $289.95 |
| | | ALL FOR | $779.85 |

| | | | |
|---|---|---|---|
| COMMODORE 64 | | | $189.95 |
| with the purchase of | | | |
| 1 | 1541 | Disk Drive | $299.95 |
| 1 | 1701 | 14" color monitor | $299.95 |
| | | ALL FOR | $789.85 |

| | | | |
|---|---|---|---|
| COMMODORE 64 | | | $189.95 |
| with the purchase of | | | |
| 1 | 1541 | Disk Drive | $299.95 |
| 1 | 1526 | Fast printer-includes IFC/cable direct connect to 64 | $349.95 |
| | | ALL FOR | $839.85 |

### COMMODORE VIC-20 $74.95*

| | | |
|---|---|---|
| | *with the purchase of | |
| 1 | Datasette program recorder | $69.95 |
| 1 | Gortek educational software | $24.95 |
| | All for | $169.85 |

| | | |
|---|---|---|
| | COMMODORE 64 Computer | $239.95 |
| 1541 | Disk drive | 279.95 |
| 1525E | Printer/direct connect ro 64 | 269.95 |
| 1626 | Printer/direct connect to 64 | 339.95 |
| 1701 | 14" color monitor | 289.95 |
| 1530 | Datasette program recorder | 79.95 |
| 1600 | Telephone modem for VIC-20/Com. 64 | 69.95 |
| 1650 | Modem, new-auto. dial/answer | 109.95 |

Software for Commodore 64 & VIC-20 at discount! Call us!

DATA 20 Accessories

| | | |
|---|---|---|
| • Z-80 card | For Commodore 64 | $269.95 |
| • 80 column card | For Commodore 64 | 169.95 |
| • 40 colomn card | For VIC-20 | 99.95 |
| • 40 column card | w/8K mem built-in for VIC-20 | 139.95 |
| • Video Pack 64K | For VIC-20 | 329.95 |

*FREE with the purchase of each of the above (*) products—Word Master/word processing software, Mailing List software & Telecommunications software.

### NEW ROYAL 64K ALPHATRONIC Computer

CP/M, color graphics, 3 video outputs—RGB for hi res 80 col color displays composite video: use with any regular monitor; modulated RF for use with your TV. Centronic printer port, 40/80 col display, KC std audio cass. port RE: $695.00 Your cost: $489.95

Optional 320 5¼" disk drive, $379.95

### TI COMPACT COMPUTER CC-40 only $199.95

Most advanced portable, fits in your briefcase!
Will perform as well as the $1000 unit!
One set of A batteries = 200 hours of operation (appx)

| | | |
|---|---|---|
| HX-1000 | Printer/plotter | $179.95 |
| HX-2000 | Wafertape (digital tape drive) | $124.95 |
| RS-232 Interface | | 89.95 |
| HX-3000 | RS-232 int. w/parallel IFC | 109.95 |

Solid state software/wafertape software-call us!

| DISKETTES | In packs of 10 | |
|---|---|---|
| Memorex 5¼" | SS/DD | $24.85 |
| Verbatim | SS/DD | 29.45 |
| Maxell | SS/DD | 33.95 |

### COMPUTER PRINTERS & MONITORS

| | | |
|---|---|---|
| Diablo 620 | Letter quality, 25 cps | $ 999.95 |
| Diablo 620 | Letter quality, 40 cps | 1799.95 |
| NEC 8023A | 100 cps with tractor | 499.95 |
| Transtar | Color printer, 30 shades, 50 cps | 499.95 |
| Okidata 82A | Graphics 120 cps bi-directional | 424.95 |
| Okidata 83A | Par/Ser., 120 cps, friction/tractor | 664.95 |
| Okidata 92P | 160 cps, graphics, bi-directional | 539.95 |
| Okidata 93P | 160 cps, graphics, frict/tract, letter quality | 889.95 |
| Star Gemini 10 | 100 cps, 2.3K buffer, Epson comp. | 369.95 |
| Star Gemini 15 | 100 cps, 15", 136 col, 2.3K buffer | 489.95 |
| Zenith | 12" green monitors, good resolution | 99.95 |
| Sanyo | 15" b/w monitor, high res-below cost! | 169.95 |
| IDS | Model 480 | 529.95 |
| IDS | 80 column color printer/all options | 1595.00 |
| IDS | 132 columns color/all options | 1695.00 |

### Texas Instruments Home Computer

TI-99/4A

16K

$99.95*

NET after $50 rebate from TI
*You pay us $149.95!*

| | |
|---|---|
| Peripheral expansion box | $199.95 |
| RS-232 card | 139.95 |
| Disk controller card | 199.95 |
| Expansion Sys. disk drive | 319.95 |
| Memory expan. card (32K) | 229.95 |
| P Code card (req. exp. card) | 199.95 |
| Telephone modem | 159.95 |
| Color monitor | 339.95 |
| Extended Basic | 79.95 |
| LOGO | 89.95 |

### HEWLETT PACKARD

| | | |
|---|---|---|
| HP-75C | Handheld comp. | $719.95 |
| HP-10C | Scientific calc | 59.95 |
| HP-11C | Scientific calc | 76.90 |
| HP-12C | Financial calc | 99.95 |
| HP-15C | Scientific calc | 99.95 |
| HP-16C | Prog. sce scien. calc | 99.95 |
| HP-97 | Prog. scien. w/print | 595.00 |
| HP-41C | Prog. calculator | 169.95 |
| HP-41CV | Prog. calculator | 219.95 |

### DON'T BUY ANY PORTABLE COMPUTER UNTIL YOU SEE THE NEW COMMODORE!

* * * COMMODORE'S SX-100 PORTABLE! * * *

*COMPLETELY PORTABLE!*

- ★ Full 64K
- ★ Color Monitor built-in
- ★ Disk drive built-in
- ★ FREE software package
- ★ Completely compatible with the Commodore 64
- ★ LOTS OF SOFTWARE AVAILABLE!

$995.00

### SANYO

NEW FABULOUS SANYO COMPUTER MBC-555 IBM PC Compatible, uses IBM software, 8088 CPU, 128K memory, 160K disk storage, color graphics, centronic port, 80 column, MS/DOS Basic, diagnostics, utilities, speaker, joystick port, word processing & spread sheet software included.
Retail: $995.00 call for best price! Optional:
128K RAM expansion; 320K to 640K drives, 8087 number cruncher, RS-232 hard disk available & more!

### SOFTWARE for ALL MICROCOMPUTERS

3000 titles available at great prices

| | |
|---|---|
| 1-2-3 Lotus for IBM | $397.00 |
| PFS: File | 114.00 |
| PFS Report | 99.95 |
| Multiplan | 219.00 |
| D Base II | 529.00 |
| Wordstar and more | 397.00 |
| Recreation— and more | |
| Zaxxon | 37.95 |
| Miner: 2049er | 36.95 |
| Frogger | 34.95 |
| Deadline | 46.95 |
| Zork II | 39.95 |
| Zork III | 39.95 |

### EPSON

MX-80FT $449.95

| | | |
|---|---|---|
| FX-80 | 160 cps | call for price |
| RX-80 | 100 cps | call for price |
| MX-100 | | call for price |

### TIMEX TS-1000 $44.95

| | |
|---|---|
| 16K RAM | 45.95 |
| Printer by Timex | 89.95 |

Call us for Timex software—buy 4 software/get 1 software free!

### brother

Personal Electronic Typewriter
EP-20 Incl. batt
FITS IN A BRIEFCASE - AC/DC
$159.95

---

## COMPUTER CASSETTES

100% Error-Free • Fully Guaranteed

| LENGTH | 12 PACK | 24 PACK |
|---|---|---|
| C-05 | 79¢ | 69¢ |
| C-10 | 89¢ | 79¢ |
| C-20 | 99¢ | 89¢ |
| Boxes | 26¢ | 21¢ |

UPS $3.00 Pkg. $18.00 Case

C-10's 39¢
(Min. 500 Case Lot)
w/labels ADD 4¢,
w/boxes ADD 13¢

FOR ORDERS ONLY
1-800-528-6050
Extension 3005

master charge VISA

**MICRO-80™ INC.**
2665-C Busby Road
Oak Harbor, WA 98277

---

## FREE

### Educational Software Catalog for Your Family

Just Released! Sunburst's new catalog, featuring the best home educational software . . . for all ages . . . selected from the nation's top publishers. Reading, math, programming, problem solving and more . . . to motivate your children and make learning come alive.

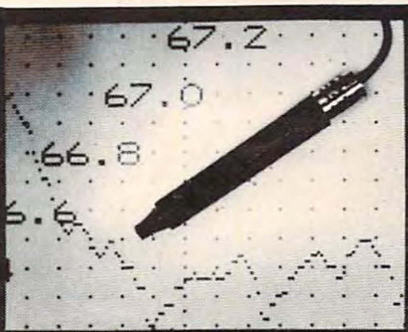
**Call Toll-Free 800-431-1934**

☐ RUSH MY FREE CATALOG

Name ______
Address ______
City ______
State ______ Zip ______
Type of computer ______

SUNBURST
Dept YK 7
39 Washington Ave.
Pleasantville, NY 10570

---



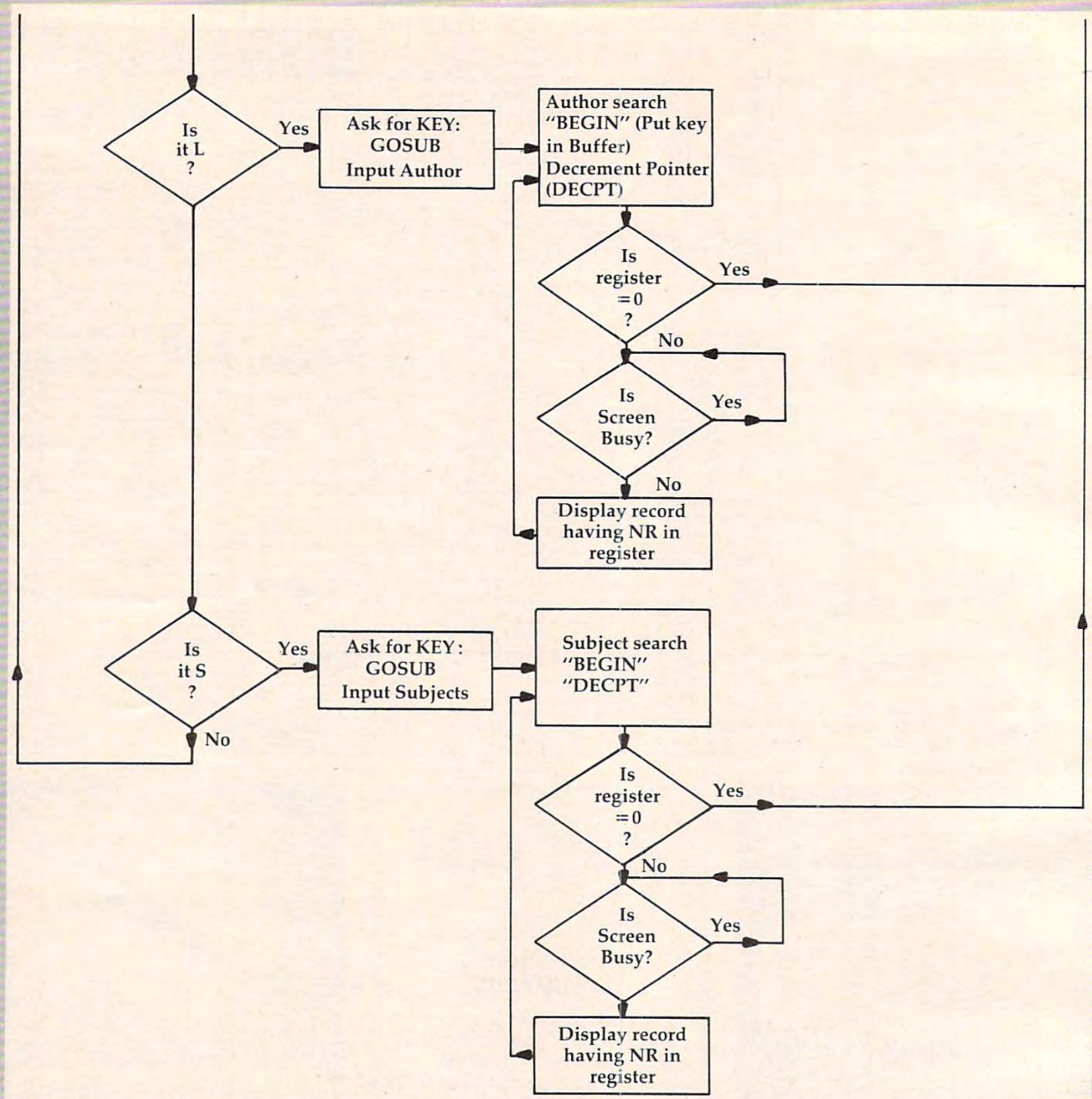

## APPLE / ATARI / COMMODORE

### MEASURE & CONTROL TEMPERATURE

DISPLAY GRAPHICS
HARD COPY OUTPUT
ALARMS & SETPOINTS
DISK FILE DATA STORAGE

**1–256 Sensors**
**Precise to 1/100 Degree**
**Complete Software**
**$129.00 Package**

**American Data Cable, Inc.**
2864 Ray Lawyer Drive, #205
P.O. Box 2212 • Placerville, CA 95667
(916) 622-3465




program, but has no function except in that system. You can simply include this program on the same disk as the main program where it is handy.

I had to create a satellite program when my main data file forced me to make a second data system. Many of the records I entered were titles of magazine or journal articles. There is no point in spelling out *The Journal of Embryology and Experimental Morphology* when everybody in the business knows it as "JEEM." Every periodical has an official abbreviation, but how to remember them all? I had to make a dictionary of journal names and their abbreviations. Of course, that meant another data file. If I had had the foresight, I could have incorporated the dictionary into the main system. Fortunately, my half-megabyte disks have lots of room, but I really do not need a second system with its files and program when it could be ancillary to the main one. Maybe you will think far enough ahead and avoid the rather clumsy solution I had to adopt.

As we've stressed throughout this series, ingenuity, careful planning, and foresight are the key ingredients to a good system. Although it may be frustrating at times, writing the system is almost as worthwhile as using it. ©

**PRODUCTS FOR ATARI* 400/800 FROM ELCOMP**

# HOFACKER

Books + Software for ATARI VIC-20 OSI SINCLAIR TIMEX

PAYMENT: check, money order, VISA, MASTER CARD, Eurocheck, ACCESS, Interbank
Prepaid orders add $3.50 for shipping (USA)
$5.00 handling for C.O.D.
All orders outside USA: add 15 % shipping. California residents add 6.5 % sales tax.

ATARI is a registered trademark of ATARI Inc.
VIC-20, CBM are registered trademarks of Commodore
APPLE is a registered trademark of APPLE Computer, Inc.


ELCOMP PUBLISHING, INC
53 Redrock Lane
Pomona, CA 91766
Phone: (714) 623 8314


## BOOKS for ATARI Computers

**ATARI BASIC – Learning by Using**
An excellent book for the beginner. Many short programs and learning exercises. All important features of the ATARI computers are described (screen drawings, special sounds, keys, paddles, joysticks, specialized screen routines, graphics, sound applications, peeks, pokes, and special stuff). Also suggestions are made that challenge you to change and write program routines.
Order #164 — $7.95

**Games for the ATARI Computer**
This book describes advanced programming techniques like player-missile-graphics and use of the hardware-registers. Contains many ready to run programs in BASIC and one called GUNFIGHT in machine language.
Order #162 — $7.95

**How to program your ATARI in 6502 Mach.Lang.**
Introduction to machine language for the BASIC programmer
Order # 169 — $9.95

**FORTH on the ATARI – Learning by Using**
Introduction, programs, applications, learning exercises
Order # 170 — $7.95
All programs from book No. 170 on disk
Order No. 7319 — $22.00 only!

**A Look into the Future – ASTROLOGY**
on your ATARI 800
How to calculate your own horoscope
Order # 171 — Incl. listing of the program — $9.95

Our catalog is free with every order. Send $1.00 and SASE for catalog only.

**SUPERMAIL**
(500 addr. on 1 disk)
Completely written in FORTH. Comes on autoboot disk. No cartridge, no DOS, no FORTH Language required!
Order# 7312 — $49.00

**SUPERINVENTORY**
(1000 items per disk)
Completely written in FORTH. Same as above. (Disk only)
Order# 7320 — $49.00

**BUSIPACK-1**
(written in FORTH). Complete order entry, inventory, mailing and invoicing. (Disk only)
Order # 7313 — $98.00

**ATAMEMO**
Datablock to keep track of your appointments. (D+C)
Order # 7310 — $29.95

**ATCASH**
Convert your ATARI 800 into a powerful cash register. (Disk only)
Order # 7307 — $49.95

**Invoicing progr. i. BASIC**
Order # 7201 (C) $29.95
Order # 7200 (D) $39.95

**Mailing List in BASIC**
Order # 7212 (C) $19.95
Order # 7213 (D) $24.95

**Inventory control in BASIC**
Order # 7214 (C) $19.95
Order # 7215 (D) $24.95

**Microcomputer Hardware Handbook** (845 pages)
Descriptions, pinouts and specifications of the most popular microprocessors and support chips.
A MUST for the hardware buff.
Order-No. 29
$14.95

## SOFTWARE IN MACHINE LANGUAGE for ATARI

**ATMONA-1**
This is a machine language monitor that provides you with the most important commands for programming in machine-language. Disassemble, dump (hex and ASCII), change memory location, block transfer, fill memory block, save and load machine-language programs, start programs. Printer option via three different interfaces.

| | | |
|---|---|---|
| Order # 7022 | cassette version | $19.95 |
| Order # 7023 | disk version | $24.95 |
| Order # 7024 | cartridge version | $59,00 |

**ATMONA-2**
This is a tracer (debugger) that lets you explore the ATARI RAM/ROM area. You can stop at previously selected address, opcode, or operand. Also very valuable in understanding the microprocessor. At each stop, all registers of the CPU may be changed. Includes ATMONA-1.

| | | |
|---|---|---|
| Order # 7049 | cassette version | $49.95 |
| Order # 7050 | disk version | $54.00 |

**ATMAS**
Macro-Assembler for ATARI-800/48k. One of the most powerful editor assemblers on the market. Versatile editor with scrolling. Up to 17k of source-Code. Very fast, translates 5k source-code in about 5 seconds. Source code can be saved on disk or cassette. (Includes ATMONA-1)

| | | |
|---|---|---|
| Order # 7099 | disk version | $89.00 |
| Order # 7999 | cartridge version | $129.00 |

**ATAS**
Same as ATMAS but without macro-capability. Cassette-based.

| | | |
|---|---|---|
| Order # 7098 | 32k RAM | $49.95 |
| Order # 7998 | 48k RAM | $49.95 |

**ATEXT-1**
This wordprocessor is an excellent buy for your money. It features screen oriented editing, scrolling, string search (even nested), left and right margin justification. Over 30 commands. Text can be saved on disk or cassette.

| | | |
|---|---|---|
| Order # 7210 | cassette version | $29.95 |
| Order # 7216 | disk version | $34.95 |
| Order # 7217 | cartridge version | $69.00 |

**GUNFIGHT**
This game (8k machine-language) needs two joysticks. Animation and sound. Two cowboys fight against each other. Comes on a bootable cassette.
Order # 7207 — $19.95

## FORTH for the ATARI

**FORTH from Elcomp Publishing, Inc.** is an extended Fig-Forth-version, Editor and I/O package included. Utility package includes decompiler, sector copy, Hexdump (ASCII), ATARI Filehandling, total graphic and sound, joystick program and player missile. Extremely powerful!
Order # 7055 — disk — $39.95

**Floating point package** with trigonometric functions (0 – 90°).
Order # 7230 — disk — $29.95

**Learn-FORTH from Elcomp Publishing, Inc.**
A subset of Fig-Forth for the beginner. On disk (32k RAM) or on cassette (16k RAM).
Order # 7053 — $19.95

## Expansion boards for the APPLE II

**The Custom Apple + Other Mysteries**
A complete guide to customizing the Apple Software and Hardware
Order-No. 680 — $24.95

We also stock the boards which are used in the book "The Custom Apple" (barebords)

| | | |
|---|---|---|
| 6522 I/O Board No. 605 | | $39.00 |
| EPROM Burner No. 607 | | $49.00 |
| 8K EPROM/RAM Board | No. 609 | $29.00 |
| Prototyping board for the Apple II | No. 604 | $29.00 |
| Slot repeater board for the Apple II | No. 606 | $49.00 |

Order two boards and get the book free!

**Care and Feeding of the Commodore PET**
Eight chapters exploring PET hardware. Includes repair and interfacing information. Programming tricks and schematics.
Order # 150 — $9.95

## Hardware – ADD-ONS for ATARI

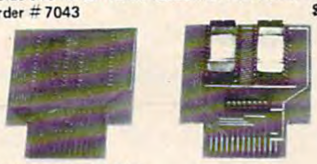
**PRINTER INTERFACE**
This construction article comes with printed circuit board and software. You can use the EPSON printer without the ATARI printer interface. (Works with gameports 3 and 4).
Order # 7211 — $19.95

**RS-232 Interface for your ATARI 400/800**
Software with connector and construction article.
Order # 7291 (5V TTL-Level) — $19.95

**EPROM BURNER for ATARI 400/800**
Works with gameports. No additional power supply needed. Comes compl. assembled with software (2716, 2732, 2532).
Order # 7042 — $179.00

**EPROM BURNER for ATARI 400/800 KIT**
Printed circuit board incl. Software and extensive construction article.
Order #7292 — $49.00

**EPROM BOARD (CARTRIDGE)**
Holds two 4k EPROMs (2532). EPROMs not included.
Order # 7043 — $29.95



**EPROM BOARD KIT**
Same as above but bare board only with description.
Order # 7224 — $14.95

## ATARI, VIC-20, Sinclair, Timex and OSI

**NEW – for your ATARI 400/800**
Astrology and Biorhythm for ATARI (Disk only!)
Order-No. 7223 — $29.95
Birth control with the ATARI (Knaus Ogino)
Order-No. 7222 — Disk only! — $29.95

**Books + Software for VIC-20 (requires 3KRAM Exp.)**
No. 4870 Wordprocessor for VIC 20, 8KRAM — $19.95
No. 4883 Mailing List for VIC-20, 16K RAM — $14.95
No. 176 Tricks for VICs (book, 115 pages) — $ 9.95

Miniassembler for VIC-20
No. 4896 — $19.95
Runfill for VIC, No. 4894 — $9.95
TIC TAC VIC, No. 4880 — $9.95
GAMEPACK I (3 Games)
No. 4881 — $14.95

**Progr. in 6502 Machine Language on your PET+CBM**
2 complete Editor/Assemblers (Source code 3 hexdump + description plus a powerful machine language monitor (Hexdump). Order-No. 166 — $19.95

**Universal Experimenter Board for the VIC-20**
(Save money with this great board). This board plugs right into the expansion slot of the VIC-20.
Order #4844 — $18.95

**Software for SINCLAIR ZX-81 and TIMEX 1000**
#2399 Machine Language Monitor — $9.95
#2398 Mailing List — $19.95

**Programming in BASIC and machine language with the ZX-81 (82) or TIMEX 1000.**
Order-No. 174 (book) $ 9.95

**BOOKS FOR OSI**
No. 157 1. Book of Ohio — $7.95
No. 158 2. Book of Ohio — $7.95
No. 159 3. Book of Ohio — $7.95
No. 160 4. Book of Ohio — $7.95
No. 161 5. Book of Ohio — $7.95

#151 8K Microsoft BASIC Ref. Man. — $9.95
#152 Expansion Handbook for 6502 and 6802 — $9.95
#153 Microcomputer Appl. Notes — $9.95

**Complex Sound Generation**
New revised applications manual for the Texas Instruments SN 76477 Complex Sound Generator.
Order # 154 — $6.95

**Small Business Programs** Order # 156
Complete listings for the business user. Inventory, Invoice Writing, Mailing List and much more. Introduction to Business Applications. — $14.90

Jim Butterfield, Associate Editor

# Bagel Break, Part 3

We've looked at some of the planning that goes into organizing a machine language game of "Bagels." Let's put the final touches together, and discuss some less obvious aspects of the way the program fits with BASIC.

We chose to start the machine language at $033C, decimal 828. The main reason for this was to make it universal – the same space is available in PET, CBM, VIC and Commodore 64 computers. It is the memory address of the cassette tape buffer (on PET/CBM, the buffer for cassette 2).

But that space is not always free and clear. If we wished to save the program to cassette tape, we might need this buffer space. The SAVE command would begin by staging the program "header" block in this area; the program would be destroyed before it was written. If we should try any BASIC 4.0 disk commands, this area would also be invaded; a simple CATALOG command would wreck our program.

For safety's sake, we should pop our machine language program into place just before we use it. What better way than to build the program as a series of BASIC DATA statements, and POKE it into its working area?

That's exactly what we do in the program here. If we examine the numbers in the DATA statements, we'll be able to spot our original program. The first two numbers, for example, are 169 and 0. These decimal numbers would translate to $A900, and that's our first instruction, LDA #$00, or, "Load the A register with the actual value of hex 00." We could trace through all of the instructions of the original program in this fashion.

## It Is BASIC

How did we get the DATA statement values in lines 100-180? We could do it by painstaking hand translation, but there are easier ways. After all, we have a computer to do the routine calculations for us. One way would be to put the hex program in place, and then write a loop using PEEK to print out the decimal values. For example,

```
FOR J = 828 TO 848: PRINT PEEK(J);:NEXT J
```

would yield a series of decimal values. Using screen editing, we could insert the commas and prefix the values with a line number and the word DATA.

Thus, we have a program that's totally BASIC. When it runs, we manufacture a machine language program and then call it. But the program handles like BASIC, lists like BASIC, and may be loaded and saved like BASIC – because it is BASIC.

A few comments on the BASIC program itself. Line 290 causes the random number generator to be scrambled, or "randomized." When we use the value zero as an argument, i.e., RND(0), the random number seed is scrambled against the clock time so that all following numbers, called with RND(1), will be unpredictable.

Lines 300 to 320 generate four random numbers, each from 65 (the ASCII letter A) to 70 (letter F) inclusive. These values are POKEd into memory for the machine language program to use.

After the call to machine language, PEEK(577) will tell us whether or not the player got the solution. Location 577 ($0241), tells us about the "exact matches": four is a correct solution, of course. If the count is less than four, we must tell the player what the solution was by PEEKing the characters back out from addresses 580 to 583 – that's where we put them.

We have looked at a simple game which uses BASIC and machine language working together. The emphasis this time was on working the problem through and commenting on the various tools that a programmer might bring to the task.

The program could well have been written entirely in BASIC. After all, Bagels doesn't need super-speed to run. But you may notice that for this sort of job, machine language brings a clean elegance to the program. The programmer often feels that machine language gives a more total control over the programming.

# MOSES™

## THE ASSEMBLER OF THE AGES
## *for the VIC 20™*

**Why MOSES?**

Programs written with **MOSES** run fifty to several hundred times faster than programs written in BASIC. A program that takes two minutes to execute in BASIC, will only take two seconds (or less) to execute when written with **MOSES.**

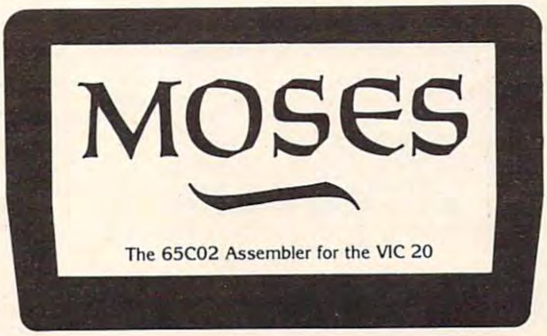
***ULTRA-FAST***
**MOSES** is written in **MACHINE LANGUAGE! MOSES** assembles dozens of times faster than assemblers written in BASIC.

***ULTRA-SOPHISTICATED***
**MOSES** makes **3 PASSES,** not just one or two like most other assemblers.

***ULTRA-FRIENDLY***
On screen menu of **EASY-TO-USE COMMANDS.**

***ULTRA-CONVENIENT***
Comes in a **CARTRIDGE** with sturdy plastic case.

**MOSES** is easy to learn and powerful. As you enter each line, **MOSES** verifies syntax, and partially assembles the line thereby reducing debugging time and conserving memory. This means an even faster assembly. Also included with **MOSES** is a machine language MONITOR with 34 powerful commands to help you debug your programs. Whether you program for fun or profit, or both, you need **MOSES.** You'll love the ability to program where imagination is your only limit.

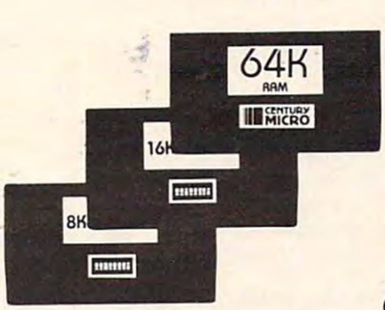



**SPECIAL OFFER!**
For a limited time, we will include a free kit to upgrade your VIC 20's outdated 6502 to the new, enhanced 65C02A microprocessor. The 65C02A is totally compatible with all VIC 20 software and hardware, uses 99% less energy, and has 27 new instructions to make programming easier and faster. And, of course, **MOSES** takes advantage of all 27 instructions. This is a $24.95 value, but is included free while quantities last.

---

# 8K • 16K
## RAM EXPANSION

A high quality memory expansion cartridge housed in a sturdy plastic case. This product has an easily accessible dip switch which gives you memory block switching ability. Use one 8K and one 16K for 24K memory or two 16K cartridges for a total of 32K memory.
**6-Month Warranty.**




# 64K
## RAM EXPANSION

It's finally here. This highly versatile ram cartridge allows you to store or write programs in two separate 32K banks that are bank selectable with software — NO SWITCHES. This product is power stingy using less than 200 mA current. A must for programmers.
**6-Month Warranty.**

---

## VIC 20 PRODUCTS:

| | |
|---|---|
| **MOSES** | $59.95* |
| **MOSES** with 8K RAM expansion | 99.95* |
| **8K RAM** Cartridge | $46.95* |
| **16K RAM** Cartridge | 69.95* |
| **64K RAM** Cartridge | 169.95* |

*Manufacturer's suggested list price

---

Call CENTURY MICRO at (916) 920-3656 for the name of a dealer in your area. Dealer inquiries invited.

1832 Tribute Rd., Suite 213, Sacramento, CA 95815

**CENTURY MICRO**

## Alspa Computer, Inc.

Price-performance leader. Includes Z80A, 8" ds/dd drives, 3 serial + 1 parallel port, winchester port, networking. Prices start below $1500. DEALER / OEM inquiries invited.

### SPECIALS on INTREGATED CIRCUITS

| | | | | |
|---|---|---|---|---|
| 6502 | 7.45 | 10/ 6.95 | 50/ 6.55 | 100/ 6.15 |
| 6502A/6512A | 8.40 | 10/ 7.95 | 50/ 7.35 | 100/ 6.90 |
| 6520 PIA | 5.15 | 10/ 4.90 | 50/ 4.45 | 100/ 4.15 |
| 6522 VIA | 6.45 | 10/ 6.10 | 50/ 5.75 | 100/ 5.45 |
| 6532 | 7.90 | 10/ 7.40 | 50/ 7.00 | 100/ 6.60 |
| 2114-L200 | | 2.45 | 25/ 2.30 | 100/ 2.15 |
| 2716 EPROM | | 4.90 | 5/ 4.50 | 10/ 4.00 |
| 2532 EPROM | | 6.90 | 5/ 6.75 | 10/ 6.45 |
| 6116 2KX8 CMOS RAM | | 6.90 | 5/ 6.75 | 10/ 6.45 |
| 4116 RAM | | 8 for 14 | | |
| Zero Insertion Force 24 pin Socket (Scanbe) | | | | 2.00 |

## Hewlett Packard

Write or call for prices.

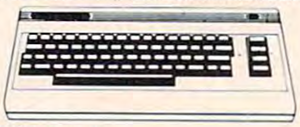
### Anchor Automation Signalman Modems

**FREE SOURCE MEMBERSHIP WITH SIGNALMAN**

All Signalman Modems are Direct Connect, and include cables to connect to your computer and to the telephone. Signalman Modems provide the best price-performance values, and start at less than $100. **Dealer and OEM inquiries invited**

| | | |
|---|---|---|
| Mark I RS232 | (99) | 59 |
| Mark II for Atari 850 | (99) | 79 |
| Mark IV for CBM/PET with software | (169) | 89 |
| Mark V for Osborne (software available) | (129) | 79 |
| Mark VI for IBM Personal Computer | (279) | 175 |
| Mark VII Auto Dial/Auto Answer | (159) | 89 |
| Mark VIII Bell 212 Auto Dial/Answer | (399) | 299 |

| | |
|---|---|
| **DC HAYES Smartmodem** | **219** |
| DC Hayes Smartmodem 1200 | 545 |



| | |
|---|---|
| **PROM QUEEN for VIC** | **170** |
| **Apple Emulator for Commodore 64** | **89** |
| **Screenmaker 80 COLUMN CARD for C64** | **145** |
| **Solid Oak 2 Level Stand for C64 or VIC** | **29** |
| C64/VIC Switch (networking) | 125 |
| **BACKUP V1.0 tape copier for C64 or VIC** | **20** |
| CARDBOARD/6 Motherboard - VIC | 64 |
| CARDAPTER/1 Atari VCS Adapter - VIC | 69 |
| CARDPRINT Printer Interface - C64/VIC | 64 |
| CARDBOARD/3s Motherboard - VIC | 32 |
| CARDCO C64/VIC Calculator Keypad | 32 |
| CARDRAM/16 RAM Expansion - VIC | 64 |
| **Complete CARDCO Line in stock** | |
| CIE and VIE IEEE Interfaces in stock | |
| BASM Compiler/Assembler for C64 | 89 |

## APPLE—FRANKLIN ITEMS

| | |
|---|---|
| **KRAFT Apple Joystick** | **40** |
| **16K RAM Card for Apple** | **59** |
| **Solid Oak 2 Level Stand for Apple** | **29** |
| Serial Card for Apple | 99 |
| **MPC RAM/80 column card for IIE (AP/TXT)** | **139** |
| Z80 Softcard and CP/M (Microsoft) | 235 |
| RANA Elite I with Controller | 389 |
| Parallel Printer Interface/Cable | 79 |
| **Apple Dumpling (Microtek) Printer Interface** | **115** |
| **Apple Dumpling with 16K Buffer** | **160** |
| Grappler + Interface | 140 |
| Kraft Products for Apple in stock | |
| DC Hayes Micromodem II | 299 |
| **PFS: File** | **100** |
| **PFS: Report** | **100** |
| Videx 80 Column Card | 209 |
| **Hayden Software for Apple 20% OFF** | |
| **PIE Writer Word Processor** | **120** |

## commodore

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

### PETSCAN I — $245 base price

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

### COMPACK/STCP — $115

**Intelligent Terminal Package for PET, CBM, C64**

Includes ACIA Hardware / STCP Software

### VE-2 IEEE to Parallel Interface — 110

Includes case, power supply, full 8-bit transmission, and switch selectable character conversion to ASCII.

**VIDEO ENHANCER for Commodore 64 — 89**

Realize video quality equal or better than composite monitor using standard color TV.

**SCREENMAKER 80 Column Adapter for C64 — 145**

Provides big screen capability for business applications.

| | | | |
|---|---|---|---|
| **VIC 20 Products** | | VIC Sargon II Chess | 32 |
| **BACKUP V1.0** | **20** | **VIC GORF** | **32** |
| VIC RAM Cards in stock | | Meteor Run (UMI) | 39 |
| VIC SuperExpander | 52 | VIC Radar Ratrace | 24 |
| VIC 16K RAM | 95 | Amok (UMI) | 20 |
| **Thorn EMI Software** | | Snakman | 15 |
| **HES Software** | | Rubik's Cube | 13 |
| VIC Omega Race | 32 | Programmers Reference | 15 |
| Spiders of Mars (UMI) | 39 | **FROGGER** | **25** |
| Programmers Aid | 45 | VIC Adventure Series | |

**VICTORY Software for VIC and C64**

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

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

| | | |
|---|---|---|
| **GRIDRUNNER (HES) VIC or C64** | | **29** |
| COCO (HES) Tutorial for C64 | | 39 |
| Gridrunner (HES) C64 or VIC | | 29 |
| Agressor (HES) | | 29 |
| HES Turtle Graphics | VIC 29 | C64 49 |
| HES Writer | VIC 30 | C64 34 |
| **Grand Master Chess (UMI) C64 or VIC** | | **27** |
| Renaissance (UMI) | VIC 29 | C64 24 |
| Synthesound (HES) | | 44 |
| HES MON | VIC or C64 | 29 |
| 6502 Professional Developement System (HES) | | 24 |
| Robot Panic 29 | Pirate's Peril 29 | Retro Ball 29 |

## DISK SPECIALS

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

### We stock VERBATIM DISKS

**Write for Dealer and OEM prices.**

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

### We stock Dysan disks

| | | | |
|---|---|---|---|
| Wabash 5" ss/sd | 10/ 1.65 | 50/ 1.60 | 100/ 1.55 |
| Wabash 5" ss/dd | 10/ 1.95 | 50/ 1.90 | 100/ 1.85 |
| Wabash 8" ss/sd | 10/ 2.00 | 50/ 1.95 | 100/ 1.85 |

### We stock MAXELL DISKS

**Write for dealer and OEM prices.**

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

**CASSETTE TAPES—AGFA PE-611 PREMIUM**

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

**DATASHIELD BACKUP POWER SOURCE — 265**

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

| | |
|---|---|
| Zenith ZVM-121 Green Phosphor Monitor | 98 |
| Zenith new color and monochrome monitors in stock | |
| **Peachtext 5000 Software Package** | **219** |
| **PFS Software for IBM and Apple in stock** | |
| VOTRAX Personal Speech System | 280 |
| VOTRAX Type-N-Talk | 160 |
| VOICE BOX Speech Synthesizer (Apple or Atari) | |
| CompuServe Subscription (5 hours free) | 32 |
| **Dynax (Brother) DX-15 Daisy Wheel Printer** | **469** |
| Prowriter Parallel Printer | 379 |
| **Panasonic 1090 Printer with Correspondence Mode** | **319** |
| USI CompuMOD 4 R F Modulator | 39 |
| Daisywriter 2000 with 48K buffer + cable | 1150 |
| Many printers available (Gemini-Star, Brother, OKI, etc.) | |
| **We Stock AMDEK Monitors** | |
| Amdek DXY-100 Plotter | 590 |
| A P Products | 15% OFF |
| Watanabe Intelligent Plotter 990 | 6-pen 1290 |
| **BROOKS 6 Outlet Surge Suppressor/Noise Filter** | **54** |
| We stock Electrohome Monitors | |
| **Synertek SYM-1 Microcomputer** | **189** |

**ALL BOOK and SOFTWARE PRICES DISCOUNTED**

| | |
|---|---|
| Panasonic 12" Monitor (20 MHz) with audio | 135 |
| Panasonic CT-160 Dual Mode Color Monitor | 259 |

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

## ZENITH data systems

| | |
|---|---|
| **HERO 1 Robot** (factory assembled) | **2145** |
| **Z29 Terminal (DEC and ADM compatible)** | **680** |
| **ZT-1 Intelligent Communications Terminal** | **369** |
| **ZT-10 Intel Terminal with Serial Port** | **340** |
| **Z100 16-bit/8-bit Systems in stock** | **CALL** |
| We stock entire Zenith line. | |

## ATARI SPECIALS

**WE STOCK ENTIRE LINE—write for prices.**
Software for Atari in stock.

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

# A B Computers

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

**OAK STAND-C64, VIC, Apple, Atari** — 29
Beautiful natural solid oak two-level stand. Rests on table above computer. Holds disk drives/cassette deck, as well as your monitor/TV.

**KMMM Pascal for PET/CBM/C64** — $79
A subset of standard Pascal with extensions. Includes Machine Language Pascal Source Editor, Machine Language P-Code Compiler, P-Code to machine language translator for optimized object code, Run-time package, Floating Point capability, User Manual, and sample programs.
Requires 32K — Please specify configuration.

**EARL for PET (disk file based)** — $65
**Editor, Assembler, Relocater, Linker**
Generates relocatable object code using MOS Technology mnemonics. Disk file input (can edit files larger than memory).

BATTERIES INCLUDED

**Paper Clip (Batteries Included)** — $109
Extremely comprehensive word processor for Commodore and Commodore 64 computers. Has features of WordPro, plus advanced functions like horizontal scroll, column move, column arithmetic, column sort, and comprehensive printer support.

**Delphi's ORACLE (Batteries Included)** — $125
Comprehensive Data Base, Report Writer, Mail Label system allowing large record size (over 8000 characters) with the number of records in a file limited only by disk capacity (7.5 MB on 9090 drive). Fast machine language routines, including full multilevel sorts.

## RAM/ROM for PET/CBM
**4K or 8K bytes of soft ROM optional battery backup.**
Use RAM/ROM as a software development tool to store data or machine code beyond the normal BASIC range, or to load a ROM image to avoid ROM socket conflicts. Possible applications include machine language sort (such as SUPERSORT), universal wedge, Extramon, etc.

| | |
|---|---|
| RAM/ROM — 4K | $75 |
| RAM/ROM — 8K | 90 |
| Battery Backup Option | 20 |

## SUBSORT for PET/CBM — $35
Excellent general purpose machine language sort routine.

**PROGRAM YOUR OWN EPROMS** — $75
Branding Iron EPROM Programmer for PET/CBM software for all ROM versions. Includes all hardware and software to program or copy 2716 and 2532 EPROMs.

**PORTMAKER DUAL RS232 SERIAL PORT** — $63
Two ports with full bipolar RS232 buffering. Baud rates from 300 to 4800. For PET/CBM, AIM, SYM.

**COMAL Package for CBM** — $25
Includes software on disk, and Comal Handbook

## SuperGraphics 2.0
**NEW Version with TURTLE GRAPHICS**
SuperGraphics, by John Fluharty, provides a 4K machine language extension which adds 35 full featured commands to Commodore BASIC to allow fast and easy plotting and manipulation of graphics and shapes on the PET/CBM video display.
SOUND commands allow you to initiate notes or songs from BASIC, and then play them in the background mode without interfering with your BASIC program.
Additionally, seven new TURTLE commands open up a whole new dimension in graphics.
Specify machine model (and size), ROM type (BASIC 3 or 4)
SuperGraphics in ROM ($A000 or $9000) — $45
Volume discounts available for schools.

## FLEX-FILE for PET/CBM Computers — NEW VERSION 2 now for C64
FLEX-FILE is a set of flexible, friendly programs to allow you to set up and maintain a data base. Includes versatile Report Writer and Mail Label routines, and documentation for programmers to use Data Base routines as part of other programs.

**RANDOM ACCESS DATA BASE**
Record size limit is 256 characters. The number of records per disk is limited only by record size and free space on the disk. File maintenance lets you step forward or backward through a file, add, delete, or change a record, go to a numbered record, or find a record by specified field (or partial field). Field lengths may vary to allow maximum information packing. Both subtotals and sorting may be nested up to 5 fields deep. Any field may be specified as a key. Sequential file input and output, as well as file output in WordPro and PaperMate format is supported. Record size, fields per record, and order of fields may be changed easily.

**MAILING LABELS**
Typical mail records may be packed 3000 per disk on 8050 (1400 in 4040). Labels may be printed any number wide, and may begin in any column position. There is no limit on the number or order of fields on a label, and complete record selection via type code or field condition is supported.

**REPORT WRITER**
Flexible printing format, including field placement, decimal justification and rounding. Define any column as a series of math or trig functions performed on other columns, and pass results such as running total from row to row. Totals, nested subtotals, and averages supported. Complete record selection, including field within range, pattern match, and logical functions can be specified.

**FLEX-FILE 2 by Michael Riley** — $110
CBM64, PET/CBM/VIC—32K, Disk. Specify configuration.

**SCREEN MAKER** (cgrs microtech) — $149
**80 Column Adapter for Commodore 64**
Expand your computer for business applications. Provides 80 column × 24 line display in a 2K video RAM. Linking software provided.

**Copy-Writer Word Processor** — $69
Full-featured professional word processor with over 800 lines of text per memory load on C64. Has features not available in many word processors such as double column printing, built in graphic capability, shorthand notations, and ability to support all printer codes.

**SPECIAL COMBINATION PACKAGE** — $200
Includes SCREEN MAKER AND Copy Writer for C64

**DISK I.C.U.** — $40
**Intensive Care Unit by L.C. Cargile**
**COMPLETE DISK RECOVERY SYSTEM FOR CBM DRIVES**
- edit disk blocks with ease
- duplicate disks, skipping over bad blocks
- complete diagnostic facilities
- unscratch scratched files
- check and correct scrambled files
- recover improperly closed files
- extensive treatment of relative files
- optional output to IEEE488 printer
- comprehensive user manual (an excellent tutorial on disk operation and theory).

Furnished on copy-protected disk with manual.
Backup disk available, $10 additional.

**CBM Software**

| | |
|---|---|
| **PAPER CLIP Word Processor** | 110 |
| **BASIC INTERPRETER for CBM 8096** | $95 |
| **PEDISK II Systems from cgrs Microtech available.** | |
| **FILEX IBM 3741/2 Data Exchange Software available.** | |
| **JINSAM Data Base Management System for CBM.** | |
| **COPY-WRITER Word Processor for PET/CBM** | $159 |
| **CASH MANAGEMENT SYSTEM** | $45 |
| Petspeed BASIC Compiler | 120 |
| Integer BASIC Compiler | 120 |
| CMAR Record Handler | 110 |
| UCSD Pascal (without board) | 135 |
| Wordcraft 80 or 8096 | 265 |

## FORTH for PET now for C64
BY L.C. Cargile and Michael Riley — $50
Features include:
- full FIG FORTH model.
- all FORTH 79 STANDARD extensions.
- structured 6502 Assembler with nested decision making macros.
- full screen editing (same as when programming in BASIC).
- auto repeat key.
- sample programs.
- standard size screens (16 lines by 64 characters).
- 150 screens per diskette on 4040, 480 screens on 8050.
- ability to read and write BASIC sequential files.
- introductory manual.
- reference manual.

For Commodore 64, or any 16K/32K PET/CBM with ROM 3 or 4, and CBM disk drive. Please specify configuration when ordering.

**Metacompiler for FORTH** — $30
Simple metacompiler for creating compacted object code which can be run independently (without FORTH system).

## PageMate 60 COMMAND WORD PROCESSOR
by Michael Riley

Paper-Mate is a full-featured word processor for Commodore computers. Page-Mate incorporates 60 commands to give you full screen editing with graphics for all 16K or 32K machines (including 8032), all printers, and disk or tape drives. Many additional features are available (including most capabilities of WordPro 3).
Page-Mate functions with all Commodore machines with at least 16K, with any printer, and either cassette or disk.
To order Page-Mate, please specify machine and ROM type.
Page-Mate (disk or tape) for PET, CBM, VIC, C64 — $40

**SM-KIT for PET/CBM** — $40
Enhanced ROM based utilities for BASIC 4. Includes both programming aids and disk handling commands.

**Commodore 64**

| | |
|---|---|
| **Hunter-Killer - Commodore 64** | 15 |
| - authentic naval warfare game (complete with sonar) | |
| Submarine Warfare (Clockwork Computers) | 29 |
| **PAPER CLIP Word Processor** | 109 |
| BASM Compiler and Assembler | 85 |
| WordPro 3+/64 | 75 |
| Vanilla PILOT with Turtle Graphics | 27 |
| - also includes sound, Toolkit, joystick support | |
| Commodore 64 Programmer Reference Guide | 16 |
| CCI Submarine Warfare | 24 |
| Laser Command | 15 |
| **EARLY GAMES for Young Children** | 25 |
| PETSPEED Compiler C64 | 120 |
| **CALC RESULT Spread Sheet Package** | 135 |
| 1000 Miles (Mille Bornes Game | 9 |
| **MicroChess** | 19 |
| Adventure (disk) | 9 |
| **PILOT 64** | 45 |
| **LOGO 64** | 45 |
| **Easy Calc 64** | 65 |
| MAE Assembler - C64 | 85 |
| Assembly Language Tutorial - C64/VIC | 27 |
| Abacus Software in stock | |
| Synthy-64 music and sound synthesizer | 26 |
| Tiny BASIC Compiler | 17 |
| ScreenGraphics-64 adds BASIC Graphics | 22 |
| Draw Poker | 5 |

VICTORY SOFTWARE®
All Victory Software in stock.

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

# A B Computers

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

## Bagels

```
90 REM BAGELS ML
100 DATA 169,Ø,141,64,2,238,64,2,173,64,2
    ,201,10,240,5,32,81,3,208,241,96
110 DATA 9,48,32,210,255,169,32,32,210,25
    5,162,Ø,142,65,2,142,66,2,142,67,2
120 DATA 32,228,255,201,65,144,249,201,71
    ,176,245,32,210,255,174,67,2
130 DATA 238,67,2,157,76,2,189,68,2,157,7
    2,2,224,3,208,223,189,72,2
140 DATA 221,76,2,208,11,238,65,2,169,Ø,1
    57,72,2,157,76,2,202,16,234,160,Ø
150 DATA 162,Ø,185,72,2,240,16,221,76,2,2
    Ø8,11,238,66,2,169,Ø,153,72,2
160 DATA 157,76,2,232,224,4,144,230,200,1
    92,4,144,223,162,Ø,169,32
170 DATA 32,210,255,189,65,2,9,48,32,210,
    255,232,224,2,144,238
180 DATA 169,13,32,210,255,173,65,2,201,4
    ,96
200 FOR J=828 TO 990
210 READ X: T=T+X
220 POKE J,X
230 NEXT J: IF T<>18169 THEN STOP
240 PRINT "BAGELS"
250 PRINT ".. GUESS MY SECRET CODE"
260 PRINT ".. I'LL TELL YOU HOW MANY"
270 PRINT ".. EXACT MATCHES AND OTHER"
280 PRINT ".. MATCHES YOU GET..."
290 X=RND(Ø)
300 FOR J=Ø TO 3
310 X=INT(RND(1)*6)+65
320 POKE 580+J,X
330 NEXT J
340 SYS 828
350 IF PEEK(577)=4 THEN PRINT "GOT IT!":G
    OTO 400
360 PRINT "THE CODE WAS: ";
370 FOR J=Ø TO 3
380 PRINT CHR$(PEEK(580+J));
390 NEXT J: PRINT
400 INPUT "WANT ANOTHER GAME";X$
410 IF X$="Y" OR X$="YES" GOTO 300
```

©

# FLOPPY DISKS SALE
# *$1.19 ea.
## Economy Model or Cadillac Quality
### *We have the lowest prices!*

**ECONOMY DISKS**

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

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

**CADILLAC QUALITY**

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

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

**100% CERTIFICATION TEST**

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

**FREE REPLACEMENT LIFETIME WARRANTY**

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

**AUTOMATIC DUST REMOVER**

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

***Loran is definitely the Cadillac disk in the world***

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

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

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

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

**RANKING FOR BEST PERFORMANCE**

| MEASUREMENTS | LORAN | Verbatim | Ultra | 3M |
|---|---|---|---|---|
| Amplitude | 1 | 2 | 3 | 3 |
| Missing bit ratio (reliability) | 1 | 3 | 3 | 2 |
| Extra bit rate | 1 | 2 | 1 | 3 |
| Modulation | 1 | 3 | 2 | 2 |
| Running torque | 1 | 1 | 1 | 2 |

(Numbers that are the same indicate a tie for that position)

***LORAN beats the competition cold!!***

Add $3.00 for postage. Add $6.00 for CANADA, PUERTO RICO, HAWAII orders. WE DO NOT EXPORT TO OTHER COUNTRIES.

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

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

C. Regena

# Playing Music On The TI

Musicians, rejoice! Here is a computer that plays music. You can play a single tone to tune an instrument or get a pitch for a song, or you can play complex rhythms with three-part harmony. Compose to your heart's content and let the computer perform your masterpiece.

Nonprofessional musicians – if you'd like to learn more about music, your computer can be a great teacher. Learn the notes on a keyboard, learn to read music, learn definitions, or learn to distinguish chords. The computer with music capabilities makes learning fun.

For those of you who prefer not to write your own programs, the TI *Music Maker* command module is available. Here's a quick review. You may compose music by choosing various types of notes or rests (quarter, eighth, half, etc.) and placing them on the staff. Choose notes for accompaniment if you wish. Build a song a measure at a time. The computer makes sure the timing works out correctly. Oh yes, you can choose your key signature, time signature, and tempo. At any time you can play or edit your composition, then save it on cassette or disk if you like.

Another section of the module is made especially for nonmusicians. You may draw lines up and down the screen at different levels for a "sound graph," then hear the computer play relational tones. Add second and third voices if you wish. This command module is really quite versatile with many options and can help you learn about music.

## CALL SOUND

To program your own music on the TI, use the CALL SOUND statement. The basic form is

**CALL SOUND(*duration, frequency, volume*)**

The *duration* is a numeric expression (number, variable, or algebraic expression which will evaluate to a number) which is the number of milliseconds you wish to play the tone. For example, 1000 would be one second. The number may be from 1 to 4250 or from -4250 to -1.

The *frequency* is a numeric expression that indicates what tone to play. The frequency is the cycles per second and may be from 110 to 44733, which is from low A on the bass staff to out-of-human-hearing range. The "Musical Tone Frequencies" table in the Appendix of the *User's Reference Guide* lists the musical notes with the corresponding frequencies. Note that you can specify numbers that are between the normal musical tones.

The *volume* is a numeric expression that indicates loudness. The volume may vary from 0 to 30, where 0 is the loudest. The volume also depends on the audio setting of your monitor or television, but you can control relative volumes of the tones with this parameter.

Try this command:

```
CALL SOUND(500,440,2)
```

The computer plays the tone of A (440) for 500 milliseconds (half a second) at a volume level of 2.

Now, if you want to tune your band instrument, just run this program.

```
100 CALL SOUND(4250,440,0)
110 GOTO 100
```

You may specify one, two, or three notes to be played in one CALL SOUND statement. Each statement has one duration, then a frequency with a volume for each note desired. Here is an example of the three notes in the C major chord:

```
CALL SOUND(1000,262,6,330,4,392,2)
```

The chord will play for 1000 milliseconds. The notes played are C at a volume 6, E at a volume 4, and G at a volume 2. Try a few chords with different frequency and volume numbers.

If you play a solo instrument, you might enjoy programming the computer to play the accompaniment chords. Tune your instrument with the computer, then you can play with the computer as your accompanist.

## Using Sheet Music

If you use three tones in the CALL SOUND statement, they may be in any order. I like to use the first frequency and volume as the melody tone, then the second and third frequencies and volumes as the accompaniment tones. This way I can keep track of which number is the melody. Also, if I start to run out of memory in a piece, I can go back to the CALL SOUND statements and delete accompaniment tones by keeping only the first frequency and volume in each statement.

You may work from a copy of written music to try out the musical capabilities of the TI. The top note is usually the melody. You may choose any two notes written directly under the melody note for the accompaniment or the other two notes in your CALL SOUND statement. To emphasize the melody, use a louder volume for the melody note and softer volumes for the accompaniment notes. For example:

```
CALL SOUND(400,262,1,196,6,159,8)
```

If you have two CALL SOUND statements together which specify the same frequencies and volumes, the notes may sound like one long note rather than two separate notes. To make the notes sound distinct, just change the volume number for one of the notes:

```
300 CALL SOUND(200,262,2,196,6,165,8)
310 CALL SOUND(200,262,3,196,6,165,8)
```

To make a bass note sound tied or held while two different melody notes are played, keep the frequency and the volume numbers the same in both statements:

```
500 CALL SOUND(300,262,2,165,8)
510 CALL SOUND(300,330,2,165,8)
```

Other statements may be executed while a note is being played. You may define graphics, draw graphics, or make calculations between CALL SOUND statements. This feature allows you to have fun choreographing pictures with music to present a musical dramatization. You do need to experiment so you don't get too many statements between the music statements or there will be gaps in the music.

A note will keep playing for its specified duration, and the computer will execute statements until either the duration runs out or another CALL SOUND statement is encountered. If another CALL SOUND statement needs to be executed, the computer waits until the first duration is finished before starting the next sound. If you prefer to have the computer go ahead with the next sound statement, use a negative number for the second statement's duration. Here is an example.

```
100 CALL SOUND(200,392,2)
110 CALL SOUND(200,330,2)
120 CALL SOUND(200,262,2)
130 CALL SOUND(200,330,2)
140 CALL SOUND(400,392,2)
150 END
```

The computer starts with the tone of G and plays for 200 milliseconds. Next the tone of E plays for 200 milliseconds, then C for 200 milliseconds, then E for 200 milliseconds, then G for 400 milliseconds. During the last note the program will end, but the note will keep playing for the 400 milliseconds.

Now change to negative durations in lines 110-140:

```
100 CALL SOUND(200,392,2)
110 CALL SOUND(-200,330,2)
120 CALL SOUND(-200,262,2)
130 CALL SOUND(-200,330,2)
140 CALL SOUND(-400,392,2)
150 END
```

This time, the computer starts by playing G. As soon as the computer comes to line 110, a CALL SOUND statement with a negative duration, the computer immediately starts the new sound – no matter what the previous duration was. Line 140 starts the sound of G as soon as the computer comes to that statement, then continues the sound for 400 milliseconds since there is not a following sound statement with a negative duration. Try running these two programs to hear the difference.

A technique I like to use in programming music is to use a variable name for the duration, and specify the numeric value of that duration variable near the beginning of the program. For example, I often use T for "tempo" or "time" or M for "metronome marking" or N for "note." If I use T to represent the duration for a quarter note, then T/2 would be an eighth note and 4*T would be a whole note. You can get exact timing in your music and let the computer calculate the durations.

*Note:* Avoiding using Q for "quarter note," especially on the TI-99/4, because the key combination of SHIFT Q is "quit." This is comparable to the FCTN (quitting on the TI-99/4A). An accidental SHIFT Q will wipe out your program and return to the title screen. With a shifted parenthesis before the variable and a shifted comma after the variable, it's too easy to get an accidental SHIFT Q.

## Variable Durations

Another advantage to using a variable duration is that you can write your song in terms of the variable, then change the tempo of the song by changing only one line (the line defining the duration) rather than each CALL SOUND statement. Here is a short example.

```
100 T=400
110 CALL SOUND(T,262,2)
120 CALL SOUND(T,294,2)
```

```
130 CALL SOUND(2*T,330,2)
140 CALL SOUND(3*T/4,349,2)
150 CALL SOUND(T/4,392,2)
160 CALL SOUND(T/2,440,2)
170 CALL SOUND(T/2,494,2)
180 CALL SOUND(T*4,523,1)
190 END
```

100 Duration of quarter note = 400
110 Quarter note
120 Quarter note
130 Half note
140 Dotted eighth note
150 Sixteenth note
160 Eighth note
170 Eighth note
180 Whole note

RUN the program. Now change line 100 to T = 800. The song is twice as long, but each note stays in the exact proportion. Change line 100 to T = 200. The song is faster, but still in proportion.

If you need to learn a song with a difficult rhythm, program the computer to play the song. Use a variable such as T for the duration. You can set the duration to a slower note, then as you learn the song you can speed it up by changing just the one line.

You may prefer to use variables for the different kinds of notes in this manner:

```
100 T=400
110 E=T/2
120 H=T*2
130 CALL SOUND(H,523,2)
140 CALL SOUND(E,494,3)
150 CALL SOUND(E,440,3)
160 CALL SOUND(T,392,2)
```

100 Quarter note duration
110 Eighth note
120 Half note

You may also want to set up a list of variables for the note names before you use them in CALL SOUND statements:

```
100 T=400
110 C=262
120 D=294
130 E=330
140 CALL SOUND(T,E,2)
150 CALL SOUND(T,D,2)
160 CALL SOUND(T,C,2)
```

You may also use a variable for the volume, such as CALL SOUND(T,D,V).

Just as in other programming, you can use FOR-NEXT loops and GOSUB and GOTO statements to help write your music. For example, if you have a musical phrase between repeat bars, you can use a FOR-NEXT loop to play it twice. If you have a common phrase used several times within a song, use a GOSUB procedure.

## Beethoven Medley

The following program, "Ludwig," illustrates the use of CALL SOUND statements to create a medley of familiar Beethoven pieces. Line 120 sets the duration of a quarter note to 400 milliseconds for the first tune, an excerpt from "Ode to Joy" of the Ninth Symphony. Lines 170-660 play this melody. In between the CALL SOUND statements are graphics statements. Lines 180-340 define graphics characters and colors, then later CALL HCHAR and CALL VCHAR statements draw a picture. The CALL SOUND statements in lines 170-400 illustrate the "tied" bass note, or a bass note held while two melody notes are played. Most of the notes are quarter notes, but line 610 has a dotted quarter note, line 650 has an eighth note, and line 660 has a half note.

Line 860 resets the duration variable T to 200 milliseconds. This time T represents an eighth note for phrases from "Ecossaises." The excerpt here is taken from music that is within repeat bars but has a first ending and a second ending. The common part of the repeat is in the subroutine at lines 1860-2230. Line 890 GOSUB 1860 plays the common phrase, then lines 920-980 play the first ending. Line 1010 repeats the common phrase with GOSUB 1860, then lines 1040-1100 contain the second ending.

Lines 1260-1420 play the third melody, "Für Elise." This example shows GOSUB commands within a FOR-NEXT loop. The subroutine for the common notes is contained in lines 2240-2420.

The final melody (lines 1430-1840) is an excerpt from the second movement of Beethoven's Fifth Symphony. Line 1430 defines the new duration T to be 800 milliseconds for an eighth note at an andante tempo. U is defined as three-fourths of an eighth note, or a dotted sixteenth note. T/4 is used for a thirty-second note. Character 128 is defined as a graphic musical note, and the embedded CALL HCHAR statements among the CALL SOUND statements place the notes on the screen.

Line 1850 (GOTO 1850) holds the picture on the screen. Press CLEAR (FCTN 4 on the TI-99/4A or SHIFT C on the TI-99/4) to stop the program.

If you prefer to save the typing time, you can obtain a copy of this program by sending $3, a stamped, self-addressed mailer, and a blank tape or disk to: REGENA, P.O. Box 1502, Cedar City, UT 84720. Please specify the name of the program.

### Ludwig

```
100 REM  BEETHOVEN MEDLEY
110 REM
120 T=400
130 CALL CLEAR
140 CALL SCREEN(3)
150 PRINT TAB(6);"BEETHOVEN MEDLEY"
    ;
160 CALL COLOR(1,2,8)
170 CALL SOUND(T,330,2,131,6)
180 CALL CHAR(96,"FFFFFFFFFFFFFFFF"
    )
```

```
190 CALL CHAR(97,"FF7F3F1F0F070301")
200 CALL SOUND(T,330,3,131,6)
210 CALL CHAR(98,"FFFEFCF8F0E0C08")
220 CALL CHAR(104,"FFFFFFFFFFFFFFFF")
230 CALL SOUND(T,349,3,131,6)
240 CALL CHAR(105,"0103070F1F3F7FFF")
250 CALL CHAR(106,"80C0E0F0F8FCFEFF")
260 CALL SOUND(T,392,2,131,6)
270 CALL CHAR(120,"00003C3E3E1E0F03")
280 CALL CHAR(121,"003878F8F8F0C0BC")
290 CALL SOUND(T,392,3,147,6)
300 CALL CHAR(113,"FF7F3F1F0F070301")
310 CALL CHAR(114,"FFFEFCF8F0E0C08")
320 CALL SOUND(T,349,3,147,6)
330 CALL COLOR(10,8,8)
340 CALL COLOR(11,3,11)
350 CALL SOUND(T,330,3,147,6)
360 CALL VCHAR(13,15,104,7)
370 CALL VCHAR(13,16,104,7)
380 CALL VCHAR(13,17,104,7)
390 CALL VCHAR(13,18,104,7)
400 CALL SOUND(T,294,3,147,6)
410 CALL HCHAR(19,14,105)
420 CALL HCHAR(19,19,106)
430 CALL HCHAR(20,13,105)
440 CALL HCHAR(20,14,104,7)
450 CALL HCHAR(20,21,106)
460 CALL SOUND(T,262,2,165,6)
470 CALL HCHAR(21,11,105)
480 CALL HCHAR(21,12,104,10)
490 CALL HCHAR(21,22,106)
500 CALL SOUND(T,262,3,165,7)
510 CALL HCHAR(22,9,105)
520 CALL HCHAR(22,10,104,14)
530 CALL HCHAR(22,24,106)
540 CALL SOUND(T,294,2,175,6)
550 CALL HCHAR(23,7,105)
560 CALL HCHAR(23,8,104,18)
570 CALL HCHAR(23,26,106)
580 CALL SOUND(T,330,2,176,5)
590 CALL CHAR(115,"80A2A2AAEEEFFFFF")
600 CALL CHAR(99,"80A2A2AAEEEFFFFF")
610 CALL SOUND(T*1.5,294,2,196,7)
620 CALL HCHAR(24,1,99,32)
630 CALL CHAR(122,"0F3F7F7D790101")
640 CALL CHAR(123,"FEFE9EC0E0E0E0E")
650 CALL SOUND(T/2,262,3,196,7)
660 CALL SOUND(2*T,262,4,165,7,131,8)
670 CALL COLOR(9,3,8)
680 CALL COLOR(10,11,8)
690 CALL HCHAR(24,6,115,22)
700 CALL HCHAR(12,15,113,2)
710 CALL HCHAR(12,17,114,2)
720 CALL HCHAR(11,13,97)
730 CALL HCHAR(11,14,96,6)
740 CALL HCHAR(11,20,98)
750 FOR I=10 TO 4 STEP -1
760 READ A,B
770 CALL HCHAR(I,A,97)
780 CALL HCHAR(I,A+1,96,B)
790 CALL HCHAR(I,A+B+1,98)
800 NEXT I
810 DATA 12,9,9,14,7,18,6,20,4,24,3,26,1,28
820 CALL HCHAR(1,1,96,94)
830 CALL HCHAR(3,31,98)
840 CALL HCHAR(2,32,32)
850 CALL HCHAR(1,32,98)
860 T=200
870 CALL COLOR(2,16,3)
880 CALL COLOR(12,3,3)
890 GOSUB 1860
900 CALL HCHAR(3,6,120)
910 CALL HCHAR(3,7,121)
920 CALL SOUND(T,466,3,117,8)
930 CALL HCHAR(4,6,122)
940 CALL HCHAR(4,7,123)
950 CALL SOUND(T,831,4,698,8)
960 CALL HCHAR(5,10,120)
970 CALL HCHAR(5,11,121)
980 CALL SOUND(T*2,831,3,698,7,233,9)
990 CALL HCHAR(6,10,122)
1000 CALL HCHAR(6,11,123)
1010 GOSUB 1860
1020 CALL HCHAR(2,15,120)
1030 CALL HCHAR(2,16,121)
1040 CALL SOUND(T,349,3,294,7,117,9)
1050 CALL HCHAR(3,15,122)
1060 CALL HCHAR(3,16,123)
1070 CALL SOUND(T,466,3)
1080 CALL HCHAR(4,26,120)
1090 CALL HCHAR(4,27,121)
1100 CALL SOUND(T*2,466,2,294,6,233,8)
1110 CALL HCHAR(5,26,122)
1120 CALL HCHAR(5,27,123)
1130 CALL COLOR(12,16,3)
1140 CALL HCHAR(6,20,120)
1150 CALL HCHAR(6,21,121)
1160 CALL HCHAR(7,20,122)
1170 CALL HCHAR(7,21,123)
1180 CALL HCHAR(8,16,120)
1190 CALL HCHAR(8,17,121)
1200 CALL HCHAR(9,16,122)
1210 CALL HCHAR(9,17,123)
1220 CALL HCHAR(2,22,120)
1230 CALL HCHAR(2,23,121)
1240 CALL HCHAR(3,22,122)
1250 CALL HCHAR(3,23,123)
1260 FOR I=1 TO 2
1270 GOSUB 2240
1280 CALL SOUND(T,415,3)
1290 CALL SOUND(T,494,3)
1300 CALL SOUND(T,523,2,110,15)
1310 CALL SOUND(T,165,4)
1320 CALL SOUND(T,220,4)
1330 CALL SOUND(T,330,3)
1340 GOSUB 2240
1350 CALL SOUND(T,523,3)
1360 CALL SOUND(T,494,4)
1370 CALL SOUND(T,440,4,110,14)
1380 CALL SOUND(T,165,10)
1390 CALL SOUND(T,220,7)
1400 NEXT I
1410 CALL SOUND(T,330,6)
1420 CALL SOUND(T*3,440,6)
```

```
1430 T=800
1440 U=T*3/4
1450 CALL CHAR(128,"080C0A0A0878F87")
1460 CALL COLOR(13,2,6)
1470 CALL SOUND(1,9999,30)
1480 CALL SOUND(U,156,6)
1490 CALL COLOR(1,2,6)
1500 CALL COLOR(9,3,6)
1510 CALL COLOR(10,11,6)
1520 CALL SOUND(T/4,208,5)
1530 CALL SOUND(T,262,3)
1540 CALL HCHAR(17,4,128)
1550 CALL SOUND(U,262,4)
1560 CALL HCHAR(15,8,128)
1570 CALL SOUND(T/4,233,4)
1580 CALL SOUND(U,208,3)
1590 CALL HCHAR(13,12,128)
1600 CALL SOUND(T/4,262,4)
1610 CALL SOUND(T+U,175,3,139,10)
1620 CALL HCHAR(13,21,128)
1630 CALL SOUND(T/4,220,3)
1640 CALL SOUND(U,233,3)
1650 CALL HCHAR(15,25,128)
1660 CALL SOUND(T/4,262,2)
1670 CALL SOUND(U,277,2,233,8)
1680 CALL HCHAR(17,29,128)
1690 CALL SOUND(T/4,262,3)
1700 CALL SOUND(U,233,2,196,8)
1710 CALL SOUND(T/4,277,2)
1720 CALL SOUND(U,196,2,156,8)
1730 CALL SOUND(T/4,233,2)
1740 CALL SOUND(U,165,3,131,8)
1750 CALL SOUND(T/4,196,3)
1760 CALL SOUND(T+U,262,2)
1770 CALL SOUND(T/4,233,4)
1780 CALL SOUND(U,220,4,175,10)
1790 CALL SOUND(T/4,175,4)
1800 CALL SOUND(U,233,2,117,10)
1810 CALL SOUND(T/4,277,3)
1820 CALL SOUND(U,196,4,156,10)
1830 CALL SOUND(T/4,156,4)
1840 CALL SOUND(2*T,208,2)
1850 GOTO 1850
1860 CALL SOUND(T,392,3,156,8)
1870 CALL HCHAR(2,2,42)
1880 CALL SOUND(T,466,3)
1890 CALL HCHAR(4,29,42)
1900 CALL SOUND(2*T,466,2,233,6,196,8)
1910 CALL HCHAR(6,14,42)
1920 CALL SOUND(T,523,3,392,6,156,8)
1930 CALL HCHAR(8,11,42)
1940 CALL SOUND(T,466,3)
1950 CALL HCHAR(2,26,42)
1960 CALL SOUND(T*2,466,2,392,6,196,8)
1970 CALL HCHAR(3,4,42)
1980 CALL SOUND(T,622,1,392,6,156,8)
1990 CALL HCHAR(2,19,42)
2000 CALL SOUND(T,466,2)
2010 CALL HCHAR(7,23,42)
2020 CALL SOUND(T*2,466,1,392,5,196,8)
2030 CALL HCHAR(3,12,42)
2040 CALL SOUND(T,523,1,392,5,156,8)
2050 CALL HCHAR(9,19,42)
2060 CALL SOUND(T,466,3)
2070 CALL HCHAR(6,7,42)
2080 CALL SOUND(T*2,466,2,392,5,196,8)
2090 CALL HCHAR(5,24,42)
2100 CALL SOUND(T,349,1,294,5,117,8)
2110 CALL HCHAR(5,17,42)
2120 CALL SOUND(T,466,3)
2130 CALL HCHAR(2,9,42)
2140 CALL SOUND(T*2,466,2,294,6,175,8)
2150 CALL HCHAR(4,20,42)
2160 CALL SOUND(T,392,2,311,5,117,8)
2170 CALL HCHAR(2,30,42)
2180 CALL SOUND(T,466,3)
2190 CALL SOUND(T*2,466,2,311,7,196,8)
2200 CALL SOUND(T,415,3,349,6,117,8)
2210 CALL SOUND(T,466,4)
2220 CALL SOUND(T*2,466,3,349,6,208,8)
2230 RETURN
2240 CALL SOUND(T,659,6)
2250 CALL SOUND(T,622,6)
2260 CALL SOUND(T,659,6)
2270 CALL SOUND(T,622,5)
2280 CALL SOUND(T,659,4)
2290 CALL SOUND(T,494,3)
2300 CALL SOUND(T,587,4)
2310 CALL SOUND(T,523,5)
2320 CALL SOUND(T,440,6,110,15)
2330 CALL SOUND(T,165,8)
2340 CALL SOUND(T,220,6)
2350 CALL SOUND(T,262,4)
2360 CALL SOUND(T,330,4)
2370 CALL SOUND(T,440,4)
2380 CALL SOUND(T,494,4,131,15)
2390 CALL SOUND(T,165,4)
2400 CALL SOUND(T,208,4)
2410 CALL SOUND(T,330,4)
2420 RETURN
2430 END
```

©

# Invisible Disk Directory For VIC And 64

Kevin E. Gough

*If you have a VIC or 64 and a 1540 or 1541 disk drive, this utility program can be very helpful. Once loaded, a simple SYS 828 will let you display your disk directory yet retain a program in memory.*

The "Invisible Disk Directory Loader" is not really invisible. It only seems to be. A BASIC program POKEs the loader into the cassette buffer as machine language. Beginning at 828 ($033C) and ending at 971 ($03CB), the loader uses 144 bytes. A knowledge of BASIC is all you need to enter and use this program.

## Using The "Invisible" Loader

If it were not for Jim Butterfield's article, "The Confusing Catalog" (COMPUTE!, March 1983), I probably would not have written the loader. I saw how easily he could load the disk directory from a program, as a file. Just OPEN 1,8,0,"$0" , input the bytes, do some manipulation, and there you have it. With the Invisible Directory Loader, you can display the directory and have any program in memory at the same time. You will no longer have to LOAD"$",8 as a program. Just type SYS 828 and press RETURN. The directory of your disk scrolls onto the screen. Use the CTRL key to slow the scroll when listing programs, or press the space bar to stop the listing.

This program will also give you the number of blocks each file uses and the number of blocks free on your disk. The directory cannot be listed on your printer.

You can also load and save cassette programs and not destroy the loader in the cassette buffer. This is because the loader also changes the start of the cassette buffer pointer at 178 ($B2) for you. Where it used to be 60 ($3C), it is now 204 ($CC), thus the buffer now starts at 972 ($03CC).

## Loading Hints

After loading a program from cassette, you will get a load error. This can be remedied with POKE 45,PEEK(174): POKE 46,PEEK(175). This indicates to the VIC or 64 the end of your program or the start of variables. Data files will not load properly with the Invisible Loader in place.

If you SYS 828 and your disk drive is not on, then nothing will happen. Turn the drive on and an error message appears on the screen. You must restore the VIC or 64 by pressing the STOP and RESTORE keys. Insert a disk, type SYS 828, RETURN, and there it is. If you do not have a disk in the drive, then the red light will flash. The screen will also scroll up with nothing on it. STOP/RESTORE, insert a disk, and type SYS 828. Before running it, be sure to save a copy to your disk or cassette. Call it "DIR".

I use this disk utility more than any other. Rarely do I LOAD "$",8. I just load "DIR",8 and run it and forget about it. It really seems invisible.

If you would rather not type in the program, I have the VIC version available. Send a blank cassette, an SASE mailer, and $3 to:

*Kevin Gough*
*24 Daisy Lane*
*Wappingers Falls, NY 12590*

### Invisible Disk Directory

```
10 I=828
20 READ A:IF A=256 THEN 40
30 POKE I,A:I=I+1:GOTO 20
40 IF PEEK(65440)=135 THEN POKE 924,189:REM
   924 HOLDS 221 ON VIC, 189 ON 64
828 DATA 169,1,32,195,255,169,36
835 DATA 141,240,3,169,48,141,241
842 DATA 3,169,1,162,8,160,0
849 DATA 32,186,255,169,2,162,240
856 DATA 160,3,32,189,255,32,192
863 DATA 255,169,64,32,144,255,162
870 DATA 1,32,198,255,32,144,255
877 DATA 32,207,255,32,207,255,32
884 DATA 207,255,32,207,255,201,0
891 DATA 240,58,32,204,255,32,228
898 DATA 255,201,32,208,3,32,196
905 DATA 3,162,1,32,198,255,32
912 DATA 207,255,168,32,207,255,72
919 DATA 152,170,104,32,205,221,169
926 DATA 32,32,210,255,32,207,255
933 DATA 201,0,208,8,169,13,32
940 DATA 210,255,76,115,3,32,210
947 DATA 255,76,162,3,169,1,32
954 DATA 195,255,32,204,255,169,204
961 DATA 133,178,96,32,228,255,201
968 DATA 32,208,249,96,256
```

©

# COMMODORE 64

## 80 COLUMN BOARD

## $149.00*


**The dream of seeing 80 columns on the screen at one time is now a reality. The Protecto Expansion Board converts your Commodore 64 to 80 columns! PLUS you get a word processor with database mailmerge, an electronic spreadsheet, and a terminal emulator. List $249. SALE $149.**

***When you buy any monitor — Commodore 64 Computer — Disk Drive - Printer! 80 Column Board purchased alone — Cost $169.00. (Disk Programs add $10.00).**

**80 COLUMN MONITOR SALE**

| | List | Sale |
|---|---|---|
| 9″ Screen - Green Text Display | $139 | $ 79 |
| 12″ Scrren - Green Text Display (anti reflective screen) | $199 | $ 99 |
| 12″ Screen - Amber Text Display (anti reflective screen) | $219 | $119 |
| 14″ Screen - Color Text Display | $299 | $269 |

**VIC 20 COMPUTER 40-80 COLUMN BOARD LIST $149.00 SALE $89.00**

- ***15 DAY FREE TRIAL***
- ***WE HAVE LOWEST COMMODORE 64 PRICES***
- ***WE HAVE OVER 500 PROGRAMS***
- ***VISA • MASTERCHARGE • COD***
- ***WE LOVE OUR CUSTOMERS***

## PROTECTO
**ENTERPRIZES** (WE LOVE OUR CUSTOMERS)

**BOX 550, BARRINGTON, ILLINOIS 60010**
**Phone 312/382-5244 to order**

# A Multicolor Atari Character Editor

Charles Brannon, Program Editor

*This program makes colorful animation easy and fun. You'll find "ANTIC Aerobics" to be an invaluable tool for working with four-color characters. There's also a submarine chase game to illustrate these techniques, an entertaining game in its own right.*

Remember the last time you played an arcade game? You probably controlled a realistic-looking ship, plane, race car, or even a Q*bert. If you tried to program such a game and ended up discouraged, here's the answer. Using one of the Atari's least used (and possibly most interesting) graphics modes, you can animate multicolor objects with simple PRINT statements.

If you've been programming for a while, you know about most of the Atari's 14 graphics modes (17 if you count GTIA modes). For example, GRAPHICS 8 is the high-resolution screen with the smallest controllable "dots." GRAPHICS 3 uses the least memory and gives you four-color graphics in a 40x24 format (each "dot" is as large as the text cursor). And there are the text modes, such as GRAPHICS 1 (double-wide), GRAPHICS 2 (double-wide and twice as high as normal text), and of course, GRAPHICS 0, the normal white-on-blue text screen.

## Silicon Symbiosis

All these graphics modes are supported by the ANTIC chip, which has been called a video microprocessor. ANTIC's job is to tell the GTIA, an essentially "dumb" chip, how to display a TV screen. Your job is to tell the ANTIC how to format a screen. Fortunately, the Atari's operating system already knows how to set up graphics screens for the ANTIC.

But this doesn't mean that you can't "do it yourself." In fact, it's rather easy to create your own custom screens with all kinds of graphics modes mixed together. Although we won't go into detail here, you can refer to Craig Chamberlain's "How to Design Custom Graphics Modes" in *COMPUTE!'s First Book of Atari Graphics* if you'd like more information.

## Hidden Modes

ANTIC can generate more graphics modes than most people think. For example, there is a special variation on GRAPHICS 0 that lets you design characters within a 9x8 matrix for true descenders (the "tail" on a g,j,p,q, or y). There's a special graphics mode "between" GRAPHICS 7 and GRAPHICS 8 that is a four-color mode with a resolution of 160x192 (some call it GRAPHICS 7½).

## ANTIC 4 And 5

However, let's limit ourselves here to the five-color character modes. In GRAPHICS 1 and 2, you get four colors of text (for example, A, a, inverse A, and inverse a). Each character can have a different color, but you can have only 64 characters, and you are limited to one color per character. But two special ANTIC modes, ANTIC 4 and ANTIC 5 (or IRG 4 and 5 according to the hardware manual), allow four colors per character.

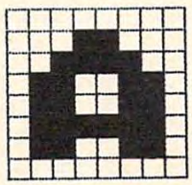
Unfortunately, the use of these modes is not intuitively obvious. It helps if you know binary (base two arithmetic). You don't really have to understand how to program characters in these modes to write games with them, as long as you have a utility to do it for you ("ANTIC Aerobics," found at the end of this article). But for those with an inclination to understand the details, the following discussion should be illuminating. Otherwise, you can skip ahead to "Using The Program."

## Assumptions

Let's start by making a few assumptions for the sake of brevity: that you understand binary numbers, know how to create custom character sets, understand the relationship between COLOR and SETCOLOR, and have a good working knowledge of BASIC.

You know that when defining a normal Atari character you get eight bits or pixels horizontally and eight bytes vertically. The letter A would be defined in binary as (we'll use open boxes for zeros, and solid squares for ones):

```
........
...OO...
..OOOO..
.OO..OO.
.OO..OO.
.OOOOOO.
.OO..OO.
........
```



Every bit represents one pixel (picture element, or "dot"). In the multicolor modes, it takes two bits to represent four colors (00,01,10,11), so the bits are "paired up." You still use only one byte per line, so you get only four pixels horizontally, although you still get eight lines vertically. Since the size of the character is the same as a GRAPHICS 0 character (in ANTIC 4), this implies that each pixel is twice as wide as a single-color pixel.

If you're using a standard character editor such as *SuperFont* or *Instedit*, you must remind yourself that you must reserve two bits per pixel.

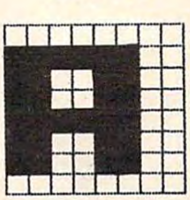
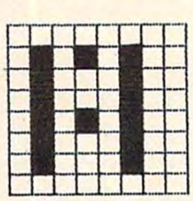
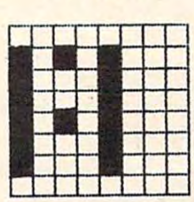
When designing a four-color character, use the following combinations:

**(00) = background color (COLOR 0, SETCOLOR 4)**
**(01) = COLOR 1 (SETCOLOR 0)**
**(10) = COLOR 2 (SETCOLOR 1)**
**(11) = COLOR 3 (SETCOLOR 2)**

In addition, if you print the character in inverse video, the COLOR 3 bit pattern (binary 11) will be displayed with the color in the fourth color register (SETCOLOR 3). Here's what three different colored A's would look like:

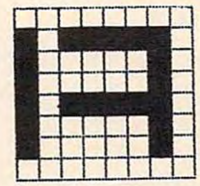
```
........      ........      ........
OOOOOO..      .O.O.O..      O.O.O...
OO..OO..      .O...O..      O...O...
OO..OO..      .O...O..      O...O...
OOOOOO..      .O.O.O..      O.O.O...
OO..OO..      .O...O..      O...O...
OO..OO..      .O...O..      O...O...
........      ........      ........
```

COLOR 3 | COLOR 2 | COLOR 1

Now there's nothing to keep you from combining all the colors in a single character. For example, here's an A with its left side in COLOR 1, the right side in COLOR 2, and the top and middle segment in COLOR 3. To the right is the same character with bit pattern 00 shown as a period, bit pattern 01 as an *, bit pattern 10 as an @, and bit pattern 11 as a #:

```
........  ----
O.OOOOO.  *##@
O.....O.  *--@
O.....O.  *--@
O.OOOOO.  *##@
O.....O.  *--@
O.....O.  *--@
........  ----
```



If you try to program other shapes, however, such as an alien invader or a race car, you will find that you need more than one character per shape. Four pixels don't give you much to work with. But if you put two characters side by side, you're back in eight-bit business. As long as you're doing that, you can create matrices of two-by-two characters, or any size you like. You can create "building block" characters, "primary" shapes that you use to build larger objects. But if you try to make larger, more complex "pictures," you'll probably discover that the task of designing each character and piecing the characters together can be rather maddening. That's where ANTIC Aerobics comes in.

## Using The Program

ANTIC Aerobics lets you draw a free-hand shape or picture that is 32 pixels wide and 16 pixels high. You can then "compile" the shape into a set of 16 characters. You display the shape as two rows of eight characters. If you put the shape into the character set on top of the alphabet, you could show it on the screen with a statement like:

```
200 PRINT "ABCDEFGH":PRINT "IJKLMNOP"
```

You can also place each shape into a string. When you PRINT the string, the shape appears. The string is made of eight characters, a cursor down, and eight cursor-lefts to back up the cursor under the first eight, then eight more characters. Following the COMPUTE! listing conventions, it would look like:

```
C$="ABCDEFGH{DOWN}{8 LEFT}GHIJKLMNOP"
```

If you also had a string filled with blanks (eight spaces, cursor down, eight cursor-lefts, and eight spaces), you could PRINT the blank string on top of the shape to blank it out. Animation made simple! Program 2 is a submarine game using shapes developed with ANTIC Aerobics. Take a look at the line-by-line explanation for more ideas on animation.

## Display List Dickering

To go into the special ANTIC modes 4 or 5, you have to change the display list. Fortunately, this is fairly simple with a mode 4 display; you just change all the 2's in the display list to 4's. You can also POKE in 5's for the double-height ANTIC 5 mode. These two lines will do either one:

ANTIC 4:

```
GRAPHICS Ø:DL=PEEK(56Ø)+256*PEEK(561)+4
FOR I=2 TO 24:POKE DL+I,4:NEXT I:POKE DL-
   1,68
```

ANTIC 5:

```
GRAPHICS Ø:DL=PEEK(56Ø)+256*PEEK(561)+4
FOR I=2 TO 12:POKE DL+I,5:NEXT I:POKE DL-
   1,69
```

You might also want to disable the cursor with POKE 752,1.

## How To Use ANTIC Aerobics

Use a joystick to draw. Press the trigger to set a point in the current color. To change colors, press either 0,1,2, or 3 (0 is used to erase). If you want to change a color, hold down SHIFT and type the number key. You will see a cursor above a 16-color bar (GTIA only; you'll see 16 densities of vertical lines if you have a CTIA, but you can still use the program). Move the joystick left or right to the color you want. Then push up or down to change the luminance (brightness). When you're through, press the trigger.

There are several other commands to make drawing easier and more fun. For example, to draw a line between two points, press P to set the first point (think "Plot"), then move the cursor to the second spot and press D ("Drawto"). If you move the cursor again and press D, another line will emanate from the original center point. If you want to draw from one line to another without having to reset the starting point, use CTRL-D. Each time you press CTRL-D, a line will be drawn from the last line. This makes it easy to draw lines at odd angles.

What if the cursor is too fast for you? The M command will give you a cursor speed from 0 (fast) to 9 (slow). Just press M and then the appropriate number key.

You can also use the insert line and delete line key (SHIFT-INSERT, SHIFT-DELETE) to insert or delete lines.

Use the S and L keys to either SAVE a shape or LOAD one previously saved. Enter the complete filename, i.e., C: for tape, or D: name for disk. If you see an error message, like "ERROR 162 ON SAVE", press a key to try again. The SAVE command will not work until you "compile" the shape with the C key (see below), since it SAVEs the character bytes, which aren't defined until you compile the shape. You can use the I (Index) command to view the disk directory. Press a key after each displayed name to view the next.

When you press C, the computer will scan the picture you've drawn and convert it into 16 characters. It will then show you what the picture would look like in ANTIC modes 4 (top) and 5 (bottom). To the right is the shape drawn in inverse video, so all bit-pair three's will be in another color (usually pink). Notice that ANTIC mode 5 has vertical pixels twice as high as ANTIC 5 – in fact, each pixel is the same size as a GRAPHICS 7 pixel. What we've got is GRAPHICS 7 resolution (or better) without the exorbitant memory consumption.

## DATA Creator

The last option lets you create DATA statements from the characters you've defined. As with the SAVE command, you can write data only after you've compiled the shape. You will be asked to choose at which line number you want to start the DATA statements and the filename for the program you want to create.

The Atari will then write a series of line numbers and DATA statements to tape or disk. To merge these lines later with your own programs, use the ENTER command (ENTER "D:name" or ENTER"C:"). You have to write the lines that READ the data and POKE it into your character set wherever you want it to go (also see Program 2). Since each shape requires 16 characters, you can fit eight shapes into one character set.

If you ever get stuck, the program has a built-in Help function that gives a quick reference list of the commands. Press H, "?", or the Help key on the 1200XL. The commands will be given one at a time at the bottom of the screen. Press a key to advance each command. When you're ready to exit the program, press CTRL-Q.

## Sub Attack Program Analysis

Here's a line-by-line explanation of "Sub Attack" (Program 2). We'll look at its structure in some detail as well as explore some programming tips and tricks.

Line 130: Lines 580-890 are the initialization routine. SUBS keeps track of how many "lives" you have. Line 575 prints from one to three miniature subs at the bottom of the screen. Each minisub is formed from custom characters.

Line 140: This line clears out applicable variables at the start of each game.

Line 150: This is part of the main loop. The hardware random number generator is used (53770) to decide on a 50/50 chance whether or not to put a ship on the screen. Ships are always spaced at least ten characters apart. Since each ship is eight characters long (including spaces), the closest two ships would be is two spaces.

Line 160: Here, one of the ships is picked. The characters for the ships are stored in a string. The statement is equivalent to R=INT(4*RND(1))* 2.

Line 170: POKE 766,1 disables cursor controls, since some of the ships contain control characters which PRINT would execute instead of display.

Line 180: This is the first line in the main loop. It continually checks to see if the high score has been topped. HSCR is initialized to 500 in line 590. If the high score is beat, a special subroutine is called, but (due to BEAT) only once per game.

Line 190: This checks for a change in the score. Many different routines add to or subtract from the score, so this one statement is responsible for noting a change from the previous score (OPTS, for "Old Points") and updating the score line. The POSITION statement centers the score.

Line 200: If there is no mine falling, and if the number in the hardware random number generator is greater than 200 (a 55/255 chance), then a mine position is chosen. If MS=0, no mine will fall. Otherwise, MS holds the offset from the upper-left corner of screen memory.

Line 210: First, we reset 766, so we can execute control characters. This one line is the core of a tricky animation technique. Instead of moving each ship by drawing and erasing, each ship is placed at the right of the screen. The CHR$(254)'s are CTRL-DELETEs, which pull the line to the left. By PRINTing two of these, we can "scroll" the line to the left. It's possible to use INSERTs to push a line to the right.

Line 220: It's preferable to use short variables like JS (joystick) and FB (fire button) than the longer statements over and over again.

Line 230: FB=0 if the button is pressed. If no torpedo is in "flight," then we set one up. If TORP is zero, no torpedo will be displayed or updated. Otherwise, TORP holds the actual screen memory location of the torpedo. FIRSTMOVE is set when the player makes any move, such as moving the sub or firing a torpedo. FIRSTMOVE is used to disable the mines falling until the player has begun to move. POKE 77,0 kills attract mode.

Line 240: JS is used as an index into an array containing -1's, 0's, and +1's. The appropriate offset (+1 for 7, right, -1 for 11, left) is added to the X (horizontal) position of the submarine. Only nonzero offsets are accepted, so the sub is always moving.

Line 250: Similar to 240, except for the vertical position of the sub. We also have a check for the START button here in case the player wants to restart the game in progress.

Line 260: If the submarine moves up or down, or changes direction, we erase the submarine before the new one is PRINTed.

Line 270: The submarine horizontal variable, SUBX, is updated and checked for wraparound. A single phrase: SUBX=32-ABS(SUBX) will reverse the illegal -1 and 32 to the legal 31 and 0.

Line 280: If the vertical variable is out of range, we just remove the offset.

Line 290: SUB$ contains the characters for both directions of the submarine. The characters include a leading space if the sub is moving right, and a trailing space if the sub is moving left. The leading or trailing space erases the previous character when the sub moves, without having to erase the whole submarine (which is somewhat "flickery").

Line 300: This is the routine for moving the falling mine. If no mine is selected, then it's skipped.

Line 310: First, we erase the previous mine (if any). The mine is two characters wide, so this makes things complicated. The next position of the mine is found by adding 40 (each screen line is 40 characters long) to the mine position. If the mine has hit the bottom of the screen, it is removed from execution (since it hasn't hit anything).

Line 320: Shorthand, again. Using LOC over and over again is shorter than using SCR+MS. SCR holds the starting address of screen memory. We "look ahead" before we POKE in the mine's characters to check for a "collision."

Line 330: If nothing is hit (PEEK returned a zero for SPACE), the mine is POKEd into its new position, and we go on to the next routine at 480 (which updates the torpedo).

Line 340: Sound effect and explosion time. We assume we've hit the submarine or a torpedo. Color register three is POKEd with random colors, causing anything on the screen drawn using this register to flash and glow. The sixtieth of a second timer is set to zero.

Line 350: A loop to wait a sixtieth of a second. Not really necessary, but we want to slow down this part because the game was compiled.

Line 360: EXL(0-3) contains the left side of three explosion scenes, and EXR(0-3) holds the right side. The sound effect is arbitrary, but the volume is stepped progressively down.

Line 370: The mine is now removed.

Line 380: If the mine hit a torpedo, then both the torpedo and the mine are removed, and the player gets 100 points.

Lines 390-410: Otherwise, the sub was hit, and we flip it back and forth to illustrate its demise.

Line 415: One less submarine, but was it the last?

Line 420: Not if this line is executed. The "dead" sub is erased, and some variables are reset. Line 575 updates the number of little submarine symbols displayed.

Line 430: Start of the "game over" section. Check for high score.

Line 440: Mode 2 without text window. Turn off display list interrupt (to be safe). Messages.

Line 450: Score line. Notice that all four colors are used, upper/lowercase, inverse and normal video.

Line 460: A loop to wait for either START or the fire button to be pressed.

Line 470: Kill attract mode again, restart the game.

Line 480: Check to see if we should move the torpedo.

Line 490: Erase the old torpedo, if indeed the torpedo was where it should be (sometimes the scroll routine will pull a ship into the space where the torpedo was).

Line 500: Move the torpedo up (minus 40 characters per line). Check to see if the torpedo has gone off the "top" of the ocean. If so, deduct ten points, but don't let the score fall below zero.

Line 510: There is no sane reason to use LOC in place of TORP, but I'd done it, and didn't want to change lines 510-550 when I realized the redundancy. Anyway, we check to see if the new position is occupied (meaning a ship). If not (=0), the torpedo is placed into the spot *if* the torpedo position is under the water still.

Line 520: The mine checks to see if it hit the torpedo, and here the torpedo checks to see if it hit either the left or the right side of the mine. If so, we just reuse part of the mine explosion routine.

Lines 530-570: A different, complex sound and explosion. The explosion moves left and right from the collision until it runs out of ship characters to blow up. It *is* complicated. The score depends on how high your ship is in the water.

Line 575: A simple FOR/NEXT loop to print from one to three "subettes" as symbols of how many lives you have left.

Line 580: The start of the really Atari-specific stuff, where characters are initialized, arrays are set up, machine language is read in, and the display list modified. Actually, line 580 is merely a useless REM statement.

Line 590: The game is not started over with RUN, since this would clear out the high score. Instead, we just make sure that we do our DIMensions only once, and then set a flag (DIMMED) to make sure it won't happen again. If you're a novice player, or a 6000-pointer, you can modify HSCR here as a goal to reach.

Line 600: We modify the display list of the 24-line GR.0 display to make it a 24-line mixed-mode display (convenient). The cursor is also turned off here.

Line 610: These SETCOLOR statements come from the ANTIC Aerobics program, which generates them with the WRITE DATA STATEMENTS option.

Line 620: The top line (DL-1) is ANTIC 5, multicolor and double-height. Lines 2 to 23 are ANTIC 4, and the last line is ANTIC 6 (a.k.a. GRAPHICS 1). Line 10 is flagged for ANTIC as where the display list interrupt will occur.

Line 630: Screen memory.

Line 640: The character set is placed eight pages (2K) behind the top of memory, about 1K beneath the screen display. The character set pointer now causes ANTIC to display *our* character set.

Line 650: This important line checks to see if the character set has already been POKEd in previously. If so, why bother to do it again?

Line 660: We put the 128-byte character set up on the screen as four rows of 32 characters so you can watch the characters as they're being redefined.

Line 670: The long list of character set data starts at 1040. Four 8x2 shapes are READ into, and thereby replace, the lowercase and graphics symbols (for this game, no big loss).

Line 680: But then we overlay most of the punctuation and math symbols with the submarine characters. We still have the alphabet, the numbers, and a few punctuation marks free.

Line 690: And here we POKE the alphabet and numbers into the character set from the default ROM set at $E000 (57344).

Lines 700-710: Here we define a couple of characters, including the small submarine used on the score line. You can use the same character set in different modes, although the multicolor characters look odd in GRAPHICS 1, and the text is hard to decipher in ANTIC 4.

Line 730: This machine language section is used for the purely cosmetic purpose of dividing the screen into two parts, sea and sky, and giving us four separate colors for each half. The display list interrupt is easy to understand. It is just a bunch of LDAs (like PEEK, Load Accumulator, a special 6502 "variable") with each color and STA (Store Accumulator into memory, like POKE) into the hardware color registers. The ANTIC chip lets us synchronize this color change with any screen line we choose, and we chose line 10 in line 620 (128 is added to the mode byte).

Line 740: We clear the screen (PUT#6,125) and draw the characters for the sun and clouds.

Line 750: We tell the operating system where our display list routine is, low byte 0, high byte 6=$0600, 1536, "page six." A single POKE to 54286 tells ANTIC to "start interrupting."

Lines 760-840: The characters for each ship, arranged as eight characters for the top half of the ship and eight for the bottom, are concatenated into a single string.

Line 850: The string holding the characters for the submarine is set up.

Lines 860-870: We READ in the +1,0, and -1 values for the joystick.

Line 880: The initial position of the submarine is set, and the explosion characters are read.

Line 890: That's it for initialization!

Line 900: DATA for the explosion characters.

Lines 910-1030: A special subroutine when you beat the high score.

Lines 1040-1846: Last, but certainly not least, the DATA statements for over 530 bytes of custom character data. This is where almost all your typing mistakes will be made.

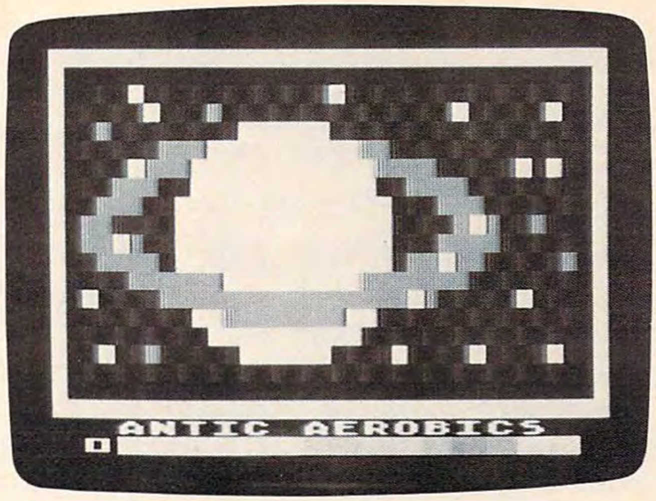
Line 1860: Here are the bytes for the small machine language display list interrupt routine. Initially, we do a store into $D40A (any write to $D40A). This makes ANTIC "hold down" the 6502's READY line, effectively freezing the microprocessor until the TV scanning beam hits the right edge of the screen. We don't want to change the colors in the middle of a line, or it would be quite jagged. This handy feature lets us wait until the beam is off the left side of the TV before we make the color change. See Program 3 for a disassembly of the display list interrupt routine (it looks long, but it's only 26 bytes).

## Program 1: ANTIC Aerobics

```
100 REM ANTIC Aerobics
110 CHSET=(PEEK(106)-8)*256
120 GOSUB 300:GOSUB 430:SPEED=4
130 LOCATE X,Y,Z
140 COLOR 1+(Z=1)
150 PLOT X,Y
160 ST=STICK(Ø)
170 IF PEEK(20)<SPEED THEN 160
180 POKE 20,Ø
190 COLOR Z:PLOT X,Y
200 IF PEEK(732) THEN POKE 732,Ø:POK
    E 764,102
210 IF PEEK(764)<255 THEN GOSUB 510
220 IF STRIG(Ø)=Ø THEN COLOR CURR:PL
    OT X,Y:LET COMPILED=Ø:IF ST=15 T
    HEN 130
230 IF ST=15 THEN 140
240 X=X+DX(ST):Y=Y+DY(ST)
250 IF X<XL THEN X=XH
260 IF X>XH THEN X=XL
270 IF Y<YL THEN Y=YH
280 IF Y>YH THEN Y=YL
290 GOTO 130
300 DIM DX(15),DY(15),ML$(20),COL(5)
    ,FN$(20)
310 CURR=1:XL=4:XH=35:YL=4:YH=19:X=X
    L:Y=YL:SX=XL:SY=YL:GOSUB 1320
320 RESTORE :FOR I=5 TO 15:READ A:DX
    (I)=A:NEXT I
330 FOR I=5 TO 15:READ A:DY(I)=A:NEX
    T I
340 DATA 1,1,1,Ø,-1,-1,-1,Ø,Ø,Ø,Ø
350 DATA 1,-1,Ø,Ø,1,-1,Ø,Ø,1,-1,Ø
360 OPEN #1,4,Ø,"K:"
370 FOR I=Ø TO 15:READ A:POKE CHSET+
    768+I,A:NEXT I
380 DATA Ø,255,240,240,240,240,255,Ø
    ,Ø,240,240,240,240,240,240,Ø
390 IF PEEK(CHSET+257)<>60 THEN FOR
    I=Ø TO 511:POKE CHSET+I,PEEK(573
    44+I):NEXT I
400 FOR I=1 TO 6:POKE CHSET+504+I,25
    5:NEXT I:POKE CHSET+504,Ø:POKE C
    HSET+511,Ø
410 IF PEEK(CHSET+784)<>17 THEN FOR
    I=1 TO 15:FOR J=Ø TO 7:POKE CHSE
    T+I*8+776+J,I+I*16:NEXT J:NEXT I
420 RETURN
430 PMBASE=(PEEK(106)-16)*256:GRID=1
440 POKE 54279,PMBASE/256
450 POKE 53277,3:POKE 559,62:POKE 62
    3,4
460 FOR I=Ø TO 3:POKE 704+I,2:POKE 5
    3248+I,64+I*32:POKE 53256+I,3:NE
    XT I
470 PØ=PMBASE+1024:BP=85:IF PEEK(PØ+
    64)=BP THEN RETURN
480 FOR I=64 TO 190 STEP 8
490 FOR J=Ø TO 7:POKE PØ+I+J,BP:POKE
     PØ+256+I+J,BP:POKE PØ+512+I+J,B
    P:POKE PØ+768+I+J,BP:NEXT J:BP=2
    55-BP
500 NEXT I:RETURN
510 GET #1,A:POKE 711,7Ø
520 IF A=ASC("W") THEN IF COMPILED T
    HEN 1810
530 IF A=87 THEN A=83:GOTO 800
540 IF A=17 THEN GRAPHICS Ø:POKE 532
    77,Ø:FOR I=Ø TO 3:POKE 53248+I,Ø
    :NEXT I:END
550 IF A=ASC("G") THEN GRID=1-GRID:F
    OR I=Ø TO 3:POKE 53248+I,(64+I*3
    2)*GRID:NEXT I:RETURN
560 IF A>47 AND A<52 THEN CURR=A-48:
    POKE 711,PEEK(707+CURR+5*(CURR=Ø
    )):RETURN
570 IF A=125 THEN GOSUB 1320:LET COM
    PILED=Ø:GOSUB 430:RETURN
580 IF A=ASC("C") THEN LET COMPILED=
    1:POP :GOTO 1020
590 IF A=7 THEN CREG=704:A=PEEK(CREG
    ):GOTO 620
600 IF A<33 OR A>35 AND A<>41 THEN 7
    40
610 CREG=708+A-33-4*(A=41):A=PEEK(CR
    EG)
620 C=INT(A/16):L=A-C*16:POKE 87,1:C
    OLOR 32:PLOT Ø,11:DRAWTO 19,11
630 POSITION C+2,11:PUT #6,95
640 T=C+DX(STICK(Ø)):IF T<Ø OR T>15
    THEN T=16-ABS(T)
650 L=L-2*DY(STICK(Ø)):IF L<Ø OR L>1
    4 THEN L=16-ABS(L)
660 A=C*16+L:POKE CREG,A:IF CREG=704
     THEN POKE 705,A:POKE 706,A:POKE
     707,A
670 IF STICK(Ø)<15 THEN POSITION C+2
    ,11:? #6;" ";:C=T
680 IF STRIG(Ø)=Ø THEN 710
690 IF PEEK(20)<SPEED THEN 680
700 POKE 20,Ø:GOTO 630
710 IF STRIG(Ø)=Ø THEN 710
720 GOSUB 1550:POSITION 3,11:? #6;"E
    ntic aerobics":POKE 711,PEEK(707
    +CURR+5*(CURR=Ø))
730 POKE 87,3:RETURN
740 IF A<>ASC("P") THEN 760
750 SX=X:SY=Y:COLOR CURR:PLOT X,Y:FO
```

```
     R W=15 TO 0 STEP -1:SOUND 0,W,12
     ,W:NEXT W:POP :LET COMPILED=0:GO
     TO 130
760  IF A=ASC("D") THEN COLOR CURR:PL
     OT SX,SY:DRAWTO X,Y:LET COMPILED
     =0:POP :GOTO 130
770  IF A=4 THEN COLOR CURR:PLOT SX,S
     Y:DRAWTO X,Y:SX=X:SY=Y:POP :LET
     COMPILED=0:GOTO 130
780  IF A=ASC("L") THEN 1560
790  IF A=ASC("S") THEN IF COMPILED T
     HEN 1450
800  IF A=ASC("S") THEN POKE 87,1:GOS
     UB 1550:POSITION 4,11:? #6;"not
     compiled":GET #1,A:GOTO 1520
810  IF A<>ASC("M") THEN 840
820  GET #1,A:IF A<48 OR A>57 THEN RE
     TURN
830  SPEED=A-48:RETURN
840  IF A<>ASC("?") AND A<>ASC("H") T
     HEN 890
850  RESTORE 1980:POKE 87,1
860  READ FN$:IF FN$="END" THEN 880
870  GOSUB 1550:POSITION 10-LEN(FN$)/
     2,11:? #6;FN$:GET #1,A:GOTO 860
880  GOTO 1520
890  IF A<>156 THEN 930
900  FOR ROW=Y*10 TO 180 STEP 10:FOR
     COL=1 TO 8:POKE SCR+ROW+COL,PEEK
     (SCR+ROW+10+COL):NEXT COL:NEXT R
     OW
910  FOR COL=1 TO 8:POKE SCR+ROW+COL,
     0:NEXT COL:LOCATE X,Y,Z
920  RETURN
930  IF A<>157 THEN 970
940  FOR ROW=190 TO Y*10+10 STEP -10
950  FOR COL=1 TO 8:POKE SCR+ROW+COL,
     PEEK(SCR+ROW-10+COL):NEXT COL:NE
     XT ROW
960  Z=0:GOTO 910
970  IF A<>ASC("I") THEN 1010
980  TRAP 1000:OPEN #2,6,0,"D:*.*":PO
     KE 87,1
990  INPUT #2,FN$:GOSUB 1550:POSITION
      1,11:? #6;FN$;:GET #1,A:GOTO 99
     0
1000 CLOSE #2:GOTO 1520
1010 RETURN
1020 FOR I=0 TO 3:POKE 53248+I,0:NEX
     T I
1030 SCR=PEEK(88)+256*PEEK(89)
1040 FOR ROW=4 TO 19
1050 FOR COL=1 TO 8
1060 LOC=SCR+ROW*10+COL:A=PEEK(LOC)
1070 POKE LOC,255-A
1080 C=COL-1:R=ROW-4:IF R>7 THEN R=R
     +56
1090 POKE CHSET+512+C*8+R,A
1100 POKE LOC,A
1110 NEXT COL:NEXT ROW
1120 FOR I=0 TO 4:COL(I)=PEEK(708+I)
     :NEXT I
1130 GRAPHICS 0:SCR=PEEK(88)+256*PEE
     K(89):DL=PEEK(560)+256*PEEK(561
     )+4
1140 POKE 752,1:POKE 756,CHSET/256
1150 FOR I=0 TO 4:POKE 708+I,COL(I):
     NEXT I:POKE 711,70
1160 POKE DL-1,4+64:FOR I=2 TO 10:PO
     KE DL+I,4:NEXT I:POKE DL+11,5:P
     OKE DL+12,5:POKE DL+13,5:POKE D
     L+14,6
1170 POKE DL+15,65:POKE DL+16,PEEK(5
     60):POKE DL+17,PEEK(561)
1180 FOR I=0 TO 1:FOR J=1 TO 8:FOR K
     =0 TO 1:FOR L=0 TO 1
1190 POKE SCR+I*40+L*10+120*K+J+284,
     63+I*8+J+L*128:NEXT L:NEXT K:NE
     XT J:NEXT I
1200 POSITION 0,13:? "PRESS fire TO
     RETURN"
1210 IF STRIG(0) THEN 1210
1220 REM RESTORE
1230 GOSUB 1320:FOR I=0 TO 4:POKE 70
     8+I,COL(I):NEXT I
1240 SCR=PEEK(88)+256*PEEK(89)
1250 FOR ROW=4 TO 19
1260 FOR COL=1 TO 8
1270 C=COL-1:R=ROW-4:IF R>7 THEN R=R
     +56
1280 A=PEEK(CHSET+512+C*8+R)
1290 POKE SCR+ROW*10+COL,A
1300 NEXT COL:NEXT ROW
1310 GOTO 130
1320 REM SET UP GR.3+16 SCREEN
1330 RESTORE 1350:FOR I=1 TO 16:READ
      A:ML$(I)=CHR$(A):NEXT I
1340 POKE 513,INT(ADR(ML$)/256):POKE
      512,ADR(ML$)-256*PEEK(513)
1350 DATA 72,169,192,141,10,212,141,
     27,208,169,10,141,26,208,104,64
1360 GRAPHICS 3+16:POKE 559,0:SCR=PE
     EK(88)+256*PEEK(89)
1370 COLOR 1:PLOT XL-2,YL-2:DRAWTO X
     H+2,YL-2:DRAWTO XH+2,YH+2:DRAWT
     O XL-2,YH+2:DRAWTO XL-2,YL-2
1380 DL=PEEK(560)+256*PEEK(561)+4
1390 POKE DL+23,6+128:POKE DL+24,2:P
     OKE 54286,192
1400 POKE 87,1:POSITION 3,11:? #6;"a
     ntic aerobics":POKE 87,3
1410 FOR I=1 TO 15:POKE SCR+244+I*2,
     97+I:POKE SCR+245+I*2,97+I:NEXT
      I:POKE SCR+244,96:POKE SCR+245
     ,97
1420 POKE 756,CHSET/256:POKE 559,62
1430 FOR I=0 TO 3:POKE 53248+I,64+I*
     32:POKE 53256+I,3:NEXT I
1440 RETURN
1450 REM SAVE ROUTINE
1460 POKE 87,1:GOSUB 1550:POSITION 0
     ,11:? #6;"s":GOSUB 1650
1470 TRAP 1500:OPEN #2,8,0,FN$
1480 FOR I=0 TO 127:PUT #2,PEEK(CHSE
     T+512+I):NEXT I:FOR I=0 TO 4:PU
     T #2,PEEK(708+I):NEXT I
1490 PUT #2,PEEK(704):CLOSE #2:GOTO
     1520
1500 GOSUB 1550:POSITION 1,11:? #6;"
     ERROR ";PEEK(195);" ON SAVE":CL
     OSE #2
1510 GET #1,A
1520 GOSUB 1550
1530 POSITION 3,11:? #6;"antic aerob
     ics"
1540 POKE 54286,192:POKE 87,3:TRAP 3
     2767:RETURN
1550 COLOR 32:PLOT 0,11:DRAWTO 19,11
     :RETURN
1560 REM LOAD ROUTINE
1570 POKE 87,1:GOSUB 1550:POSITION 0
     ,11:? #6;"l":GOSUB 1650
1580 TRAP 1620:OPEN #2,4,0,FN$
1590 FOR I=0 TO 127:GET #2,A:POKE CH
```

```
     SET+512+I,A:NEXT I:FOR I=Ø TO 4
     :GET #2,A:COL(I)=A:NEXT I
1600 GET #2,A:FOR I=Ø TO 3:POKE 704+
     I,A:NEXT I
1610 CLOSE #2:POKE 54286,192:POP :TR
     AP 32767:LET COMPILED=1:GOTO 12
     2Ø
1620 GOSUB 155Ø:POSITION 1,11:? #6;"
     ERROR ";PEEK(195);" ON LOAD":CL
     OSE #2
1630 GET #1,A
1640 GOTO 152Ø
1650 REM FILENAME INPUT
1660 POSITION 1,11:? #6;"fn";CHR$(15
     9);
1670 ZL=1
1680 POSITION 4+ZL,11:PUT #6,223
1690 GET #1,A
1700 IF A=155 THEN 179Ø
1710 IF A=126 THEN IF ZL>1 THEN ZL=Z
     L-1:COLOR 32:PLOT 5+ZL,11:GOTO
     168Ø
1720 IF NUM AND (A<48 OR A>57) THEN
     169Ø
1730 IF NUM=Ø AND ZL=1 AND A<65 OR A
     >9Ø THEN 169Ø
1740 IF A=42 OR A=46 OR A=58 THEN 17
     6Ø
1750 IF (A<48 OR A>57) AND (A<65 OR
     A>9Ø) THEN 169Ø
1760 IF ZL=15 THEN 169Ø
1770 POSITION 4+ZL,11:PUT #6,A:FN$(Z
     L)=CHR$(A):ZL=ZL+1
1780 GOTO 168Ø
1790 NUM=Ø:IF ZL=1 THEN POP :GOTO 15
     2Ø
1800 FN$=FN$(1,ZL-1):RETURN
1810 REM WRITE DATA
1820 POKE 87,1:GOSUB 155Ø:POSITION 1
     ,11:? #6;"ln";CHR$(159);:NUM=1:
     GOSUB 167Ø
1830 LN=Ø:FOR I=1 TO LEN(FN$):A=ASC(
     FN$(I))-48:IF A>=Ø AND A<1Ø THE
     N LN=LN*1Ø+A:NEXT I
1840 IF I<LEN(FN$) THEN POP
1850 GOSUB 155Ø:GOSUB 165Ø
1860 TRAP 195Ø:OPEN #2,8,Ø,FN$
1870 PRINT #2;LN;" ";:FOR I=Ø TO 4:A
     =PEEK(7Ø8+I):C=INT(A/16):L=A-C*
     16
1880 PRINT #2;"SE.";I;",";C;",";L;:I
     F I<4 THEN PUT #2,58
1890 NEXT I:PRINT #2:LN=LN+1Ø
1900 FOR I=Ø TO 127 STEP 8
1910 PRINT #2;LN;" DATA ";
1920 FOR J=Ø TO 7:PRINT #2;PEEK(CHSE
     T+512+I+J);:IF J<7 THEN PUT #2,
     44
1930 NEXT J:PRINT #2:LN=LN+1Ø:NEXT I
1940 TRAP 32767:CLOSE #2:GOTO 152Ø
1950 GOSUB 155Ø:POSITION 1,11:? #6;"
     #";PEEK(195);" ON WRITE"
1960 GET #1,A
1970 CLOSE #2:GOTO 152Ø
1980 DATA C :COMPILE
1990 DATA 1-4 :COLOR
2000 DATA shift 1-4 :SETCOLOR
2010 DATA S :SAVE
2020 DATA L :LOAD
2030 DATA I :DISK INDEX
2040 DATA W :WRITE DATA STMTS
2050 DATA M :MOTION (Ø-9)
```



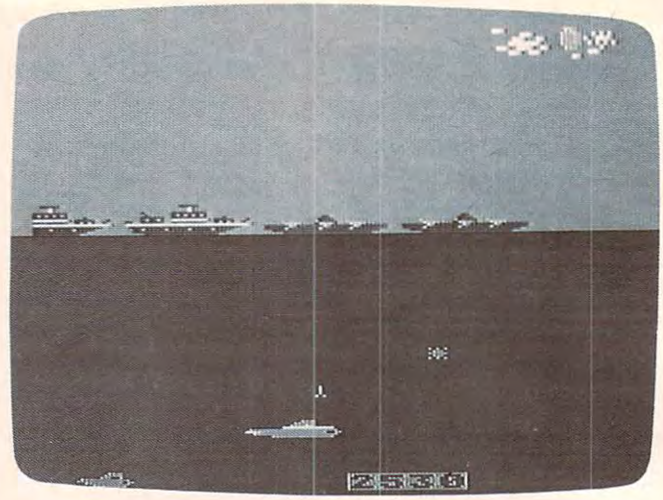

*A shape resembling the planet Saturn being edited with the ANTIC Aerobics Editor.*

```
2060 DATA P :PLOT
2070 DATA D :DRAWTO
2080 DATA ctrl D :DRAWTHRU
2090 DATA G : GRID ON/OFF
2100 DATA ctrl G:GRID COLOR
2110 DATA shift INSERT:LINE
2120 DATA shift DELETE:LINE
2130 DATA ctrl Q :QUIT
2140 DATA END
```

## Program 2: Sub Attack – An Example Game

```
100 REM  SUB ATTACK
110 REM
120 REM
130 GOSUB 58Ø:SUBS=3:GOSUB 575:REM N
    UMBER OF LIVES
140 PTS=Ø:BEAT=Ø:DX=Ø:OPTS=PTS:FIRST
    MOVE=Ø
150 X=1:IF PEEK(5377Ø)<128 THEN 18Ø
160 R=INT(4*PEEK(5377Ø)/256)*2
170 POKE 766,1:POSITION 31,8:? SHIP$
    (R*8+1,R*8+8):POSITION 31,9:? SH
    IP$(R*8+9,R*8+16)
180 IF PTS>HSCR AND BEAT=Ø THEN GOSU
    B 91Ø
190 IF PTS<>OPTS THEN POSITION 1Ø-LE
    N(STR$(SCORE))/2,23:? PTS;"  ";:
    OPTS=PTS
200 IF MS=Ø AND FIRSTMOVE AND PEEK(5
    377Ø)>2ØØ THEN MS=INT(3Ø*PEEK(53
    77Ø))/256+4ØØ
210 POKE 766,Ø:POSITION Ø,9:? CHR$(2
    54);CHR$(28);CHR$(254);:X=X+1:IF
     X=1Ø THEN 15Ø
220 JS=STICK(Ø):FB=STRIG(Ø)
230 IF FB=Ø AND TORP=Ø THEN TORP=SCR
    +SUBY*4Ø-4Ø+SUBX+7*(DX=1):FIRSTM
    OVE=TORP:POKE 77,Ø
240 IF DX(JS)<>Ø THEN DX=DX(JS):POKE
     77,Ø
250 DY=DY(JS):IF PEEK(53279)=6 THEN
    1ØØ
260 IF DY OR DX<>ODX THEN POSITION S
    UBX,SUBY:? "{8 SPACES}";:ODX=DX:
    FIRSTMOVE=DX:POKE 77,Ø
270 SUBX=SUBX+DX:IF SUBX<Ø OR SUBX>3
    1 THEN POSITION SUBX-DX,SUBY:? "
    {8 SPACES}";:SUBX=32-ABS(SUBX)
280 SUBY=SUBY+DY:IF SUBY<11 OR SUBY>
```

```
    22 THEN SUBY=SUBY-DY
290 P=1+8*(DX<1):POSITION SUBX,SUBY:
    ? SUB$(P,P+7);"{LEFT}";
300 IF MS=Ø THEN 48Ø
310 POKE SCR+MS,Ø:POKE SCR+MS+1,Ø:MS
    =MS+4Ø:IF MS>919 THEN MS=Ø:GOTO
    48Ø
320 LOC=SCR+MS:P1=PEEK(LOC):P2=PEEK(
    LOC+1)
330 IF P1=Ø AND P2=Ø THEN POKE LOC,1
    Ø2:POKE LOC+1,1Ø3:GOTO 48Ø
340 FOR V=12 TO Ø STEP -3:FOR I=Ø TO
     2:POKE 711,PEEK(5377Ø):POKE 2Ø,
    Ø
350 IF PEEK(2Ø)<1 THEN 35Ø
360 POKE LOC,EXL(I):POKE LOC+1,EXR(I
    ):SOUND Ø,7Ø,8,V:SOUND 1,PEEK(53
    77Ø),Ø,V:NEXT I:NEXT V
370 POKE LOC,Ø:POKE LOC+1,Ø
380 IF P1=112 OR P2=112 THEN MS=Ø:TO
    RP=Ø:PTS=PTS+1ØØ:GOTO 18Ø
390 FOR I=15Ø TO Ø STEP -5:POSITION
    SUBX,SUBY:? SUB$(FL*8+1,FL*8+8);
    :FL=1-FL
400 FOR J=I TO 1 STEP -2Ø
410 SOUND Ø,J/1Ø,8,I/1Ø:NEXT J:NEXT
    I
415 SUBS=SUBS-1:IF SUBS=Ø THEN 43Ø
420 POSITION SUBX,SUBY:? "
    {8 SPACES}";:SUBX=16:SUBY=22:DX=
    Ø:DY=Ø:MS=Ø:GOSUB 575:GOTO 18Ø
430 IF PTS>HSCR THEN HSCR=PTS
440 GRAPHICS 18:POKE 54286,64:SETCOL
    OR 4,9,14:POSITION 5,Ø:? #6;"GAM
    E OVER"
450 POSITION 5,5:? #6;"SCORE:";PTS:P
    OSITION 5,11:? #6;"press start"
460 IF PEEK(53279)<>6 AND STRIG(Ø) T
    HEN 46Ø
470 POKE 77,Ø:GOTO 1ØØ
480 IF TORP=Ø THEN 18Ø
490 IF PEEK(TORP)=112 THEN POKE TORP
    ,Ø
500 TORP=TORP-4Ø:IF TORP<SCR+36Ø THE
    N TORP=Ø:PTS=(PTS-1Ø)*(PTS>1Ø):G
    OTO 18Ø
510 LOC=TORP:IF PEEK(LOC)=Ø THEN POK
    E LOC,112*(TORP>SCR+4ØØ):GOTO 18
    Ø
520 IF PEEK(TORP)=1Ø2 OR PEEK(TORP)=
    1Ø3 THEN LOC=SCR+MS:P1=112:GOTO
    34Ø
530 FOR V=14 TO Ø STEP -2:FOR I=2 TO
     3:L=Ø:R=Ø
540 FOR UP=Ø TO 4Ø STEP 4Ø:POKE 711,
    PEEK(5377Ø)
550 POKE LOC-L-UP,EXL(I):POKE LOC+R-
    UP,EXR(I):A1=(PEEK(LOC-L-1)<>Ø):
    A2=(PEEK(LOC+R+1)<>Ø):NEXT UP:L=
    L+A1:R=R+A2
560 SOUND Ø,L+R,Ø,V:IF A1 OR A2 THEN
     54Ø
570 NEXT I:NEXT V:TORP=Ø:PTS=PTS+4Ø+
    (23-SUBY)*5:GOTO 18Ø
575 POSITION 1,23:? "{7 SPACES}";:FO
    R I=1 TO SUBS:POSITION I*2-1,23:
    ? "[\";:NEXT I:RETURN
580 REM Initialization
590 IF  NOT DIMMED THEN DIM DX(15),D
    Y(15),SHIP$(128),SUB$(16),EXL(3)
    ,EXR(3),MSG$(1ØØ):LET DIMMED=1:H
    SCR=5ØØ
```



*An example of edited graphics in the Atari game "Sub Attack."*

```
600 GRAPHICS Ø:DL=PEEK(56Ø)+256*PEEK
    (561)+4:POKE 752,1
610 SETCOLOR Ø,11,4:SETCOLOR 1,Ø,12:
    SETCOLOR 2,1,1Ø:SETCOLOR 3,4,6:S
    ETCOLOR 4,1Ø,8
620 POKE DL-1,69:FOR I=2 TO 23:POKE
    DL+I,4:NEXT I:POKE DL+1Ø,128+4:P
    OKE DL+24,6
630 SCR=PEEK(88)+256*PEEK(89)
640 CHSET=(PEEK(1Ø6)-8)*256:POKE 756
    ,CHSET/256
650 IF PEEK(CHSET+2Ø)=85 THEN 74Ø
660 FOR I=Ø TO 3:FOR J=Ø TO 31:POKE
    SCR+I*4Ø+8Ø+2+J,I*32+J:NEXT J:NE
    XT I
670 RESTORE 1Ø4Ø:FOR I=512 TO 1Ø23:R
    EAD A:POKE CHSET+I,A:NEXT I
680 FOR I=Ø TO 127:READ A:POKE CHSET
    +I,A:NEXT I
690 FOR I=128 TO 511:POKE CHSET+I,25
    5-PEEK(57344+I):NEXT I
700 FOR I=Ø TO 7:POKE CHSET+2Ø8+I,25
    5:NEXT I:POKE CHSET+214,239
710 FOR I=Ø TO 15:READ A:POKE CHSET+
    472+I,A:NEXT I
730 FOR I=Ø TO 25:READ A:POKE 1536+I
    ,A:NEXT I
740 PUT #6,125:POSITION 3Ø,Ø:FOR I=Ø
     TO 7:PUT #6,I:NEXT I
750 POKE 512,Ø:POKE 513,6:POKE 54286
    ,192
760 RESTORE 77Ø:FOR I=1 TO 64:READ A
    :SHIP$(I)=CHR$(A):NEXT I
770 DATA 16Ø,16Ø,16Ø,16Ø,16Ø,16Ø,16Ø
    ,16Ø
780 DATA 8,9,1Ø,11,12,13,14,15
790 DATA 32,32,18,19,2Ø,21,22,23
800 DATA 24,25,26,27,28,29,3Ø,31
810 DATA 32,97,98,99,1ØØ,1Ø1,32,32
820 DATA 1Ø4,1Ø5,1Ø6,1Ø7,1Ø8,1Ø9,11Ø
    ,111
830 DATA 16Ø,16Ø,16Ø,16Ø,16Ø,16Ø,16Ø
    ,16Ø
840 DATA 12Ø,121,122,123,124,125,32,
    32
850 FOR I=1 TO 16:SUB$(I)=CHR$(I+31)
    :NEXT I:SUB$(16)=CHR$(32)
860 FOR I=5 TO 15:READ A:DX(I)=A:REA
    D A:DY(I)=A:NEXT I
```

```
870 DATA 1,1,1,-1,1,0,0,0,-1,1,-1,-1
    ,-1,0,0,0,0,1,0,-1,0,0
880 SUBX=16:SUBY=22:FOR I=0 TO 3:REA
    D A,B:EXL(I)=A:EXR(I)=B:NEXT I
890 RETURN
900 DATA 246,247,208,209,254,255,0,0
910 HSCR=PTS:MSG$="::::::::::::::::
    :::":POKE 711,90
920 MSG$(21)="CONGRATULATIONS:::new
    :::high:::score::::":MSG$(LEN(MS
    G$)+1)=STR$(PTS)
930 MSG$(LEN(MSG$)+1)=":::::::::::::
    ::::::::":CX=31
940 FOR I=1 TO LEN(MSG$)-20
950 SOUND 0,10,8,8
960 POSITION 1,23:? MSG$(I,I+17);:PO
    KE 20,0
970 IF PEEK(20)<2 THEN 970
980 SOUND 0,30,8,8
990 POSITION CX,2:? "qrstu ";:POKE 2
    0,0:CX=CX-1:IF CX=0 THEN POSITIO
    N CX,2:? "{7 SPACES}";:CX=31
1000 IF PEEK(20)<2 THEN 1000
1010 NEXT I:BEAT=1
1020 COLOR 32:PLOT 0,23:DRAWTO 18,23
     :PLOT 18,23:PLOT 0,2:DRAWTO 39,
     2:SOUND 0,0,0,0:GOTO 575
1030 RETURN
1040 REM  GAME CHARACTERS FOLLOW 
1050 DATA 168,0,0,0,42,0,0,0
1060 DATA 0,2,42,170,10,168,0,0
1070 DATA 0,160,2,170,160,42,0,0
1080 DATA 0,0,128,0,160,0,0,0
1090 DATA 15,63,63,63,15,0,0,0
1100 DATA 192,240,242,240,192,0,160,0
1110 DATA 0,34,10,136,2,168,0,0
1120 DATA 0,32,160,136,0,0,0,0
1130 DATA 0,0,0,0,85,85,5,0
1140 DATA 0,0,192,63,95,117,85,21
1150 DATA 0,0,0,192,85,85,85,85
1160 DATA 1,5,53,31,87,93,85,85
1170 DATA 64,80,84,244,213,85,85,85
1180 DATA 0,0,48,15,87,93,85,85
1190 DATA 0,0,0,240,213,85,85,85
1200 DATA 0,0,0,0,85,85,80,0
1210 DATA 128,8,3,143,3,128,8,0
1220 DATA 2,48,240,252,224,200,0,128
1230 DATA 0,0,0,0,0,0,5,7
1240 DATA 0,0,8,85,42,42,85,119
1250 DATA 0,130,24,64,0,0,84,116
1260 DATA 0,2,33,0,0,0,0,0
1270 DATA 0,2,24,145,2,0,0,0
1280 DATA 0,4,0,32,80,0,0,0
1290 DATA 0,0,21,10,5,0,0,0
1300 DATA 0,0,85,170,117,85,21,5
1310 DATA 7,10,85,170,215,85,85,85
1320 DATA 119,170,85,170,93,85,85,85
1330 DATA 116,170,85,170,117,85,85,85
1340 DATA 0,0,85,170,215,85,85,85
1350 DATA 0,0,84,170,85,84,80,64
1360 DATA 0,0,0,128,0,0,0,0
1370 DATA 0,0,0,0,0,0,0,0
1380 DATA 0,0,0,0,0,0,64,16
1390 DATA 0,0,0,0,0,0,0,0
1400 DATA 0,0,0,5,0,0,85,119
1410 DATA 0,0,0,84,128,128,85,119
1420 DATA 0,0,0,0,0,0,64,64
1430 DATA 0,4,1,5,1,4,0,0
1440 DATA 0,68,80,84,80,68,0,0
1450 DATA 0,0,80,170,21,5,5,0
1460 DATA 85,117,85,170,85,213,85,21
1470 DATA 80,208,80,170,85,93,85,85
1480 DATA 85,170,85,170,85,85,85,85
1490 DATA 85,170,85,170,85,213,85,85
1500 DATA 64,129,65,170,85,93,85,85
1510 DATA 16,85,221,170,85,85,80,0
1520 DATA 0,1,4,170,80,0,0,0
1530 DATA 0,16,16,16,16,136,0,0
1540 DATA 1,0,0,5,31,31,5,0
1550 DATA 84,5,4,85,213,86,85,0
1560 DATA 0,80,0,80,85,89,84,0
1570 DATA 0,0,0,1,85,64,0,0
1580 DATA 4,17,4,84,64,0,0,0
1590 DATA 32,2,51,15,131,8,128,0
1600 DATA 34,0,200,240,194,48,0,8
1610 DATA 0,0,0,0,5,1,0,0
1620 DATA 0,0,0,2,85,93,85,5
1630 DATA 20,20,20,170,85,93,85,85
1640 DATA 20,20,20,170,85,93,85,85
1650 DATA 0,0,0,128,85,93,85,84
1660 DATA 0,0,0,0,84,64,0,0
1670 DATA 0,128,8,32,15,131,3,2
1680 DATA 0,0,194,192,242,252,192,50
1690 DATA 0,0,0,0,0,0,0,0
1700 DATA 0,0,0,0,5,42,1,0
1710 DATA 0,0,0,0,85,165,85,0
1720 DATA 0,0,0,3,85,85,85,85
1730 DATA 0,5,21,255,85,85,85,85
1740 DATA 85,85,105,255,85,85,85,85
1750 DATA 0,0,0,255,85,84,85,85
1760 DATA 0,0,0,192,80,20,80,64
1770 DATA 0,0,0,3,5,20,5,1
1780 DATA 1,1,1,255,85,21,85,85
1790 DATA 84,85,165,255,85,85,85,85
1800 DATA 0,64,80,255,85,85,85,85
1810 DATA 0,0,0,240,85,85,85,85
1820 DATA 0,0,0,0,85,90,85,64
1830 DATA 0,0,0,0,80,168,64,0
1840 DATA 0,0,0,0,0,0,0,0
1845 DATA 0,0,1,63,127,31,7,0
1846 DATA 120,248,200,254,251,254,25
     2,0
1850 REM Type the following carefull
     y--Machine Language!
1860 DATA 72,169,6,141,10,212,141,22
     ,208,169,40,141,23,208,169,10,1
     41,24,208,169,128,141,26,208,10
     4,64
1870 END
```

## Program 3:
### Disassembly Of The Display List Interrupt Routine

| | | | |
|---|---|---|---|
| 72 | PHA | | ; Since this is an interrupt, we want to save any registers we use so that when we return from the interrupt, the original routine won't notice anything. PHA means to "push" the accumulator onto the stack. The stack will hold the previous value in the accumulator until we "pull" it off. |
| 169 6 | LDA | #6 | ;Grey (0*16+6) |
| 141 10 212 | STA | $D40A | ;WSYNC (wait for synchronization) |
| 141 22 208 | STA | $D018 | ;Color register zero (hardware) |
| 169 40 | LDA | #40 | ;2*16+8, light orange |
| 141 23 208 | STA | $D019 | ;Color register one. Since ML is so fast, the TV beam still hasn't reappeared. We'll be able to make all our changes without having to store to WSYNC again. |
| 169 10 | LDA | #10 | ;Light white |
| 141 24 208 | STA | $D01A | ;Color register two |
| 169 128 | LDA | #128 | ;8*16+0, dark blue, for the ocean |
| 141 26 208 | STA | $D01C | ;Background color register |
| 104 | PLA | | ;Restore accumulator |
| 64 | RTI | | ;Return from Interrupt (like RETURN from a GOSUB) |

©

# High Speed Mazer

Gary E. Marsa

*This update of previously published "Maze Generator" uses machine language to construct a random maze in less than two seconds – for PET, VIC, and 64. Also, there's "Munchmaze," a fast-action strategy game to show off the maze utility, with versions for the PET and 64.*

If you tried Charles Bond's "Maze Generator" in the December 1981 COMPUTE!, you'll remember how fascinating it was to watch the maze being constructed on the screen right before your eyes. It's a clever program and lacks only one thing – speed. It takes my PET about 38 seconds to construct a full-screen maze. After watching it make several mazes, it occurred to me that a machine language version would be much faster.

The machine language maze generator was written on an Upgrade PET, and conversions for Original ROMs and 4.0 ROMs were incorporated into the loader program (Program 1). Also included are versions for the VIC-20 (Program 2) and the 64 (Program 3).

The PET version uses 176 bytes and will fit into one of the cassette buffers. It uses the second cassette buffer, but 4.0 BASIC users may prefer to use the first cassette buffer. If so, change the value of S in line 120 to 634. If you would like to use one or both buffers for utility programs, instructions for loading machine language into high RAM are given at the end of the loader (lines 450 on).

The VIC-20 version occupies 201 bytes and must be loaded into high memory because it's too large for the cassette buffer. The extra bytes in this version are needed to handle color. Screen and border are both white, and the maze color is chosen randomly. All colors except black and white are used.

While typing in the loader program, make special note of DATA items beginning with an asterisk (*) or a plus sign (+). Be sure to include these symbols. When you've finished typing, be *sure* to SAVE the program before RUNning it. When the program is RUN, it first POKEs the machine language into memory and then offers a demonstration. Mazes will be constructed on your screen as long as you keep pressing keys.

## Speeding Up The Maze

Converting Charles Bond's algorithm from BASIC to machine language was accomplished by a nearly line-by-line translation of the original BASIC program. Although the machine language program executes far faster than the original BASIC program, the maze does not appear on the screen instantaneously. But the motion is so fast it's hard to follow with your eyes. I timed the PET maze construction at 1.65 seconds. The VIC and 64 versions take about half as much time.

The mazes are 39 columns by 23 rows on the PET and 64, and 21 columns by 21 rows on the VIC. These are maximum sizes. Changing the maze dimensions is possible, but not particularly easy, especially if you want to center the maze on the screen. You can try this:

| | | |
|---|---|---|
| **PET & 64:** | **POKE S+37,C** | (where C is >10 or <40) |
| | **POKE S+53, R** | (where R is >10 or <24) |
| **VIC:** | **POKE S+62, C** | (where C is >10 or <22) |
| | **POKE S+78, R** | (where R is >10 or <22) |

C is the number of columns, R the number of rows, and S the SYS address *minus eight*. C and R *must be odd numbers*. Mazes smaller than the maximum size will not be centered, but will start in the upper-left-hand corner of the screen.

## Munchmaze

Shortly after converting the maze generator to machine language, I wrote a machine language game called "Munchmaze," in which a character hurries through the maze dropping bread crumbs as it goes. You move your character around with the appropriate keys and try to munch as many of the bread crumbs as you can before the character catches you. The game ends when the two characters collide or when you accumulate 10,000 points.

There are three speed levels: slow, moderate, and fast. Both characters move at the same speed, but the computer character beats you on the corners. Also, you have to change directions manu-

# It's time for your computer to grow up.

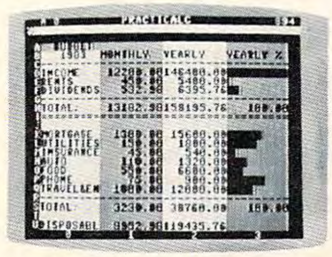
**Meet PractiCalc.™ The world's most versatile spreadsheet at only $40.***

Games are fun when it's time to play. But at heart, your Commodore 64™ or VIC-20™ is a full-grown computer straining to gallop at the touch of your fingers.

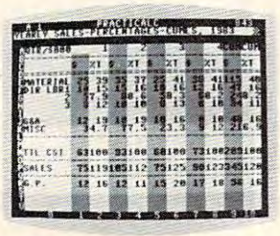
Commodore 64 and VIC-20. Not just for games...

But the problem is, most software's been designed for toys. Not for real computers. Until now.

Now comes PractiCalc. And there's not much in the world of grown-up computing it won't do. It'll keep addresses (in alphabetical or numerical order), phone numbers (likewise), make budgets and menus, project profits, keep track of expenses, inventories, investments, what have you. For your home, your school, your business.



Budgets, inventories, projections, you name it...



Spreadsheet analysis...

PractiCalc gives you everything you'd expect from a spreadsheet, like adding and subtracting, multiplying and dividing, calculating square roots, logarithms, exponential numbers and even trig functions. And PractiCalc gives you a lot more.

Want to make a chart? Hit a key and the high or low resolution graphics of PractiCalc Plus or PractiCalc 64 will turn numbers into graphs. It'll even print out everything that shows on the screen.

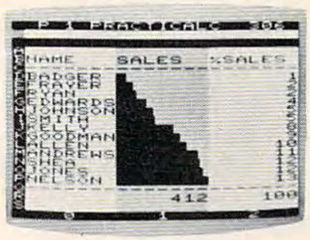
High or low resolution graphics...

Want a lot of facts and figures at your fingertips? PractiCalc stores more than 2,000 cells in up to 100 columns and 250 rows. It'll sort them, search them, or shuffle them for you.

High-speed and wild card search...

You can add entries, delete them, or move them around—numerically or alphabetically, and instantly, of course. And you'll find PractiCalc unusually friendly, flexible, and forgiving.

All of which is a far cry from just playing with a joystick. Still, PractiCalc has one thing in common with a computer game. The price. Only $40.



Alpha or numeric sort...

**PRACTICALC**™ by **COMPUTER SOFTWARE ASSOCIATES, INC.**

Exclusively distributed by **Micro Software International, Inc.**
The Silk Mill, 44 Oak Street, Newton Upper Falls, MA 02164 • (617) 527-7510

*$40 is suggested retail price for tape version of PractiCalc-20 ($45 for disk version; PractiCalc Plus or PractiCalc 64 = $50 for tape version, $55 for disk).

ally; it doesn't. The computer moves its character according to the same "left-turn rule" used by the mouse in Charles Bond's original maze generator program. You must be aware of this in order to find temporary hiding places.

There's another tricky feature, too. Sometimes, when the two characters are moving from opposite directions toward each other, the computer character goes right on by and no collision occurs. Just breathe a sigh of relief and continue munching – you were lucky.

The maze in Munchmaze is not constructed on the screen, but in another area of RAM. It is then transferred to the screen, where the maze appears all at once; then there is a one-second delay before the action begins. If you break out of the program for any reason just type SYS 12311 to restart.

Programs 4 and 5 are versions of Munchmaze for 4.0 and Upgrade PETs, respectively. Program 6 is a 64 version of Munchmaze.

If you would rather not type these programs yourself, I'll make copies for you. Send a self-addressed, stamped mailer, a blank cassette, and $3 to:

*Gary Marsa*
*320 Terrace, Apt. 2-S*
*Flushing, MI 48433*

I have available the Maze Generator for all PETs and the VIC, and Munchmaze for Original, Upgrade or 4.0 PETs. Please tell me which version(s) you want.

## Special Note To 8032 And Fat Forty Owners

Because of keyboard differences between "old style" 40-column PETs and "Fat Forties," Munchmaze will not work properly on Fat Forties, or 8032s. Your machine is a "Fat Forty" if a bell rings when you turn it on.

Munchmaze 4.0 will work properly on these computers if you type in these two lines instead of the ones that appear in the listings:

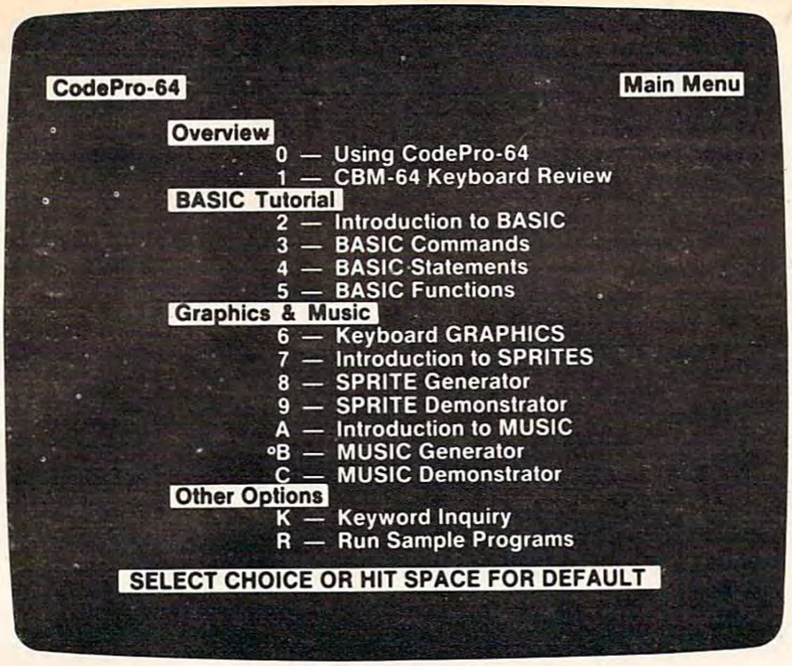
```
13314 DATA 255,255,255,40,0,182
13320 DATA 184,180,178,160,32,58
```

## Program 1:
### Maze Generator For 40-Column PET With Original, Upgrade, Or 4.0 ROMs

```
50 REM FOR 40-COLUMN PET/CBMS WITH ORIGI
   NAL, UPGRADE, OR 4.0 ROMS
100 CLR: POKE 59468,12: X=RND(-TI)
110 P=PEEK(50003): Z=84-82*(P=0)
120 S=826: A=S
130 PRINT "{CLR}{2 DOWN}LOADING...
    {2 DOWN}"
140 READ X$: IF X$="XXX" THEN 200
150 R=ASC(X$): Q=VAL(MID$(X$,1-(R<48)))
160 IF R=42 THEN X=Z+Q: GOTO 190
170 IF R<>43 THEN X=Q: GOTO 190
180 Y=S+Q: X=INT(Y/256): Y=Y-256*X: POKE
     A,Y: A=A+1
190 POKE A,X: A=A+1: GOTO 140
200 IF P=0 THEN POKE S+63,69: POKE S+66,
    222
210 IF P=160 THEN POKE S+63,41: POKE S+6
    4,210
220 PRINT "ACTIVATE WITH {RVS}SYS"; S+8
230 PRINT "{2 DOWN}PRESS ANY KEY FOR DEM
    ONSTRATION MAZES."
240 PRINT "{2 DOWN}PRESS 'Q?' WHEN YOU WA
    NT TO QUIT.{3 DOWN}": GOTO 260
250 SYS S+8: PRINT "{HOME}PRESS KEY..."
260 GET X$: IF X$="" THEN 260
270 IF X$<>"Q" THEN 250
280 DATA 1, 0, 216, 255, 255, 255, 40, 0
    , 169, 81
290 DATA 133, *0, 169, 40, 133, *2, 169,
     128, 133, *1
300 DATA 133, *3, 169, 147, 32, 210, 255
    , 162, 0, 160
310 DATA 0, 169, 160, 145, *2, 200, 192,
     39, 208, 249
320 DATA 24, 165, *2, 105, 40, 133, *2,
    144, 2, 230
330 DATA *3, 232, 224, 23, 208, 229, 160
    , 0, 169, 4
340 DATA 145, *0, 32, 127, 223, 165, 140
    , 41, 3, 133
350 DATA 1, 170, 10, 168, 24, 185, +0, 1
    01, *0, 133
360 DATA *4, 185, +1, 101, *1, 133, *5,
    24, 185, +0
370 DATA 101, *4, 133, *2, 185, +1, 101,
     *5, 133, *3
380 DATA 160, 0, 177, *2, 201, 160, 208,
     18, 138, 145
390 DATA *2, 169, 32, 145, *4, 165, *2,
    133, *0, 165
400 DATA *3, 133, *1, 76, +62, 232, 138,
     41, 3, 197
410 DATA 1, 208, 189, 177, *0, 170, 169,
     32, 145, *0
420 DATA 224, 4, 240, 26, 138, 10, 168,
    162, 2, 56
430 DATA 165, *0, 249, +0, 133, *0, 165,
     *1, 249, +1
440 DATA 133, *1, 202, 208, 238, 76, +62
    , 96, XXX
450 REM MAKE THESE ADDITIONS & CHANGES T
    O LOAD MACHINE CODE INTO HIGH RAM:
460 REM
470 REM{2 SPACES}70 P=PEEK(50003): M=52-
    82*(P=0)
480 REM{2 SPACES}80 Y=PEEK(M)+256*PEEK(M
    +1)-177: X=INT(Y/256): Y=Y-256*X
490 REM{2 SPACES}90 POKE M,Y: POKE M+1,X
    : POKE M-4,Y: POKE M-3,X
500 REM{2 SPACES}110 P=PEEK(50003): Z=84
    -82*(P=0): M=52-82*(P=0)
510 REM{2 SPACES}120 S=PEEK(M)+256*PEEK(
    M+1): A=S
```

## Program 2: Maze Generator For VIC

```
10 REM MAZE (VIC)
20 REM MAZE GENERATOR IN MACHINE LANGUAG
   E
```




# NEW! For the Commodore 64™ ANNOUNCING

# CodePro-64™

## A new concept in interactive visual learning . . .

Now you can learn to code in BASIC and develop advanced programming skills with graphics, sprites and music—**visually.** You learn by interacting with CodePro-64, a new concept in interactive visual learning.

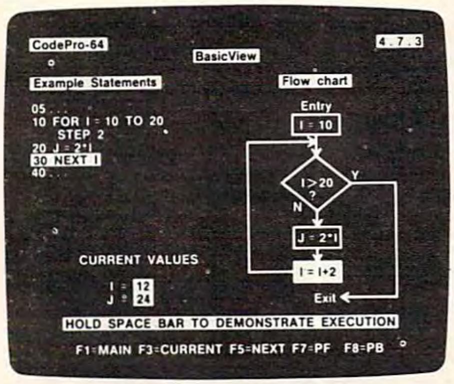
### SEE PROGRAM EXECUTION

Imagine actually seeing BASIC statements execute. CodePro-64 guides you through structured examples of BASIC program segments. You enter the requested data or let CodePro-64 do the typing for you. (It will not let you make a mistake.)

After entering an example you invoke our exclusive **BasicView™** which **shows you how the BASIC program example executes.**

You step through and actually see the execution of sample program statements by simply pressing the space bar. CodePro-64 does the rest.

You see statements with corresponding **flow chart graphics** and variable value displays. You learn by visual examples.




### EXTENSIVE TUTORIAL

CodePro-64's extensive tutorial guides you through each BASIC command, program statement, and function. You get clear explanations. Then you enter program statements as interactive examples. Where appropriate, you invoke BasicView to see examples execute and watch their flow charts and variables change.

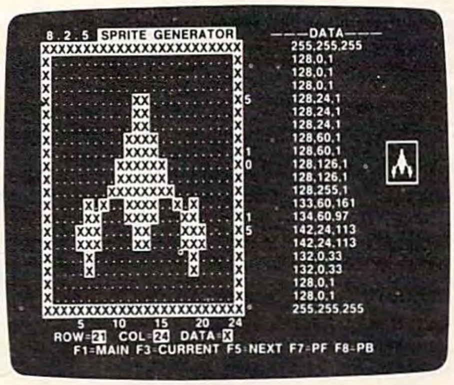
By seeing graphic displays of program segment execution you learn by visual example. You **learn faster and grasp programming concepts easier** with CodePro-64 because you immediately see the results of your input.

You control your learning. You can go through the tutorial sequentially, or return to the main menu and select different topics, or **use keywords** to select language elements to study. You can page back and forth between screens within a topic at the touch of a function key.

CodePro-64 **lets you follow your interests** and practice with interactive examples. But you can never get "lost". F1 will always return you to the main menu. Once you have practiced and mastered the BASIC language elements you move on to more advanced concepts. You learn about sprite and music programming.

### SPRITE GENERATOR & DEMONSTRATOR

CodePro-64's sprite generator lets you **define your own sprites** on the screen. You learn how to define sprites and what data values correspond to your sprite definitions. (You can then use these values to write your own programs.) You can **easily experiment** with different definitions and make changes to immediately see the effects.

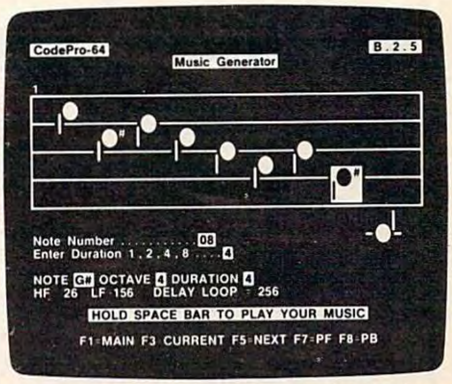



We also help you learn to program with sprites by giving you a **sprite demonstrator** so you can see the effect of changing register values. You can experiment by moving your sprite around in a screen segment, change its color or priority, and see the effects of your changes. You learn by visual examples.

### MUSIC GENERATOR & DEMONSTRATOR

To teach you music programming CodePro-64 gives you an interactive music generator and demonstrator. First we help you set all your SID parameters (attack/decay, sustain/release, waveform, etc.). Then you enter notes to play and **we show your tune graphically as it plays,** note by note, on the scale. You learn by seeing and hearing the results of your input.

> **OUR GUARANTEE**
>
> **We guarantee your satisfaction.** You must be satisfied with CodePro-64 for the Commodore-64. Try it for 10 days and if for any reason you are not satisfied return it to us (undamaged) for a full refund. **No risk.**




Our music demonstrator **lets you experiment** with various combinations of music programming parameters and hear the results. You can **quickly modify any of the SID register values** to hear the effects of the change. For example, you could easily change waveform and attack/decay values while holding all other SID values constant. By seeing your input and hearing the result you quickly learn how to create new musical sounds and special sound effects.

### AND MORE . . .

We don't have enough space to tell you everything CodePro-64 offers. You need to see for yourself. BASIC tutorials, graphics, sprites, music, keyboard review, sample programs—the main menu shown above gives you just a summary of the contents of this powerful educational product.

Whether you're a beginning programmer or an experienced professional, CodePro-64 will help you improve your Commodore 64 programming skills. We're sure because CodePro-64 was developed by a team of two professionals with **over 25 years** of software development experience.

CodePro-64 is a professional quality educational program for the serious student of personal computing. And it's **fully guaranteed.** Order yours today.

### HOW TO ORDER

Order your copy of CodePro-64 today by mail or phone. Send only $59.95 plus $3.00 shipping and handling to:

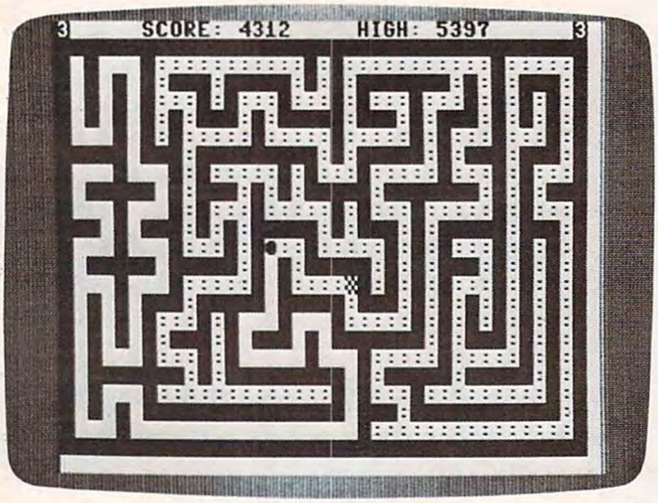
**SYSTEMS MANAGEMENT ASSOCIATES**
3700 Computer Drive, Dept. C
Raleigh, N.C. 27609

Please **specify tape or diskette** version. MasterCard/VISA accepted. For faster service on credit card orders call (919)-787-7703.

Commodore 64 is a trademark of Commodore Business Machines, Inc.

Ad no. 733, Copyright 1983, SMA

**Dealer inquiries invited.**




*A game of "Munchmaze" being played on the 64.*

```
50 REM FOR THE VIC-20 (ANY MEMORY SIZE)
100 Y=PEEK(55)+256*PEEK(56)-202: X=INT(Y
    /256): Y=Y-256*X
110 POKE 55,Y: POKE 56,X: POKE 51,Y: POK
    E 52,X
120 CLR: POKE 36879,27: PRINT CHR$(142);
    : X=RND(-TI)
130 S=PEEK(55)+256*PEEK(56): A=S
140 PRINT "{CLR}{2 DOWN}LOADING..."
150 READ X$: IF X$="XXX" THEN 200
160 R=ASC(X$): Q=VAL(MID$(X$,1-(R<48)))
170 IF R<>43 THEN X=Q: GOTO 190
180 Y=S+Q: X=INT(Y/256): Y=Y-256*X: POKE
     A,Y: A=A+1
190 POKE A,X: A=A+1: GOTO 150
200 PRINT "{HOME}": IF PEEK(210)<>16 THE
    N 220
210 POKE S+17,16: POKE S+45,148: POKE S+
    48,149
220 PRINT "{DOWN}ACTIVATE WITH"
230 PRINT "{2 SPACES}{RVS}SYS"; S+8
240 PRINT "{2 DOWN}PRESS ANY KEY FOR"
250 PRINT "DEMONSTRATION MAZES."
260 PRINT "{2 DOWN}PRESS 'Q' WHEN YOU"
270 PRINT "WANT TO QUIT.": GOTO 290
280 SYS S+8: PRINT "{HOME}PRESS KEY..."
290 GET X$: IF X$="" THEN 290
300 IF X$<>"Q" THEN 280
310 PRINT "{CLR}": POKE 36879,27
320 DATA 1, 0, 234, 255, 255, 255, 22, 0
    , 169, 45
330 DATA 133, 87, 169, 22, 133, 89, 169,
     30, 133, 88
340 DATA 133, 90, 169, 25, 141, 15, 144,
     32, 95, 229
350 DATA 32, 148, 224, 165, 143, 41, 7,
     201, 2, 48
360 DATA 245, 160, 0, 153, 0, 150, 153,
     0, 151, 200
370 DATA 208, 247, 162, 0, 160, 0, 169,
     160, 145, 89
380 DATA 200, 192, 21, 208, 249, 24, 165
    , 89, 105, 22
390 DATA 133, 89, 144, 2, 230, 90, 232,
     224, 21, 208
400 DATA 229, 160, 0, 169, 4, 145, 87, 3
     2, 148, 224
410 DATA 165, 143, 41, 3, 133, 1, 170, 1
     0, 168, 24
420 DATA 185, +0, 101, 87, 133, 91, 185,
      +1, 101, 88
430 DATA 133, 92, 24, 185, +0, 101, 91,
     133, 89, 185
440 DATA +1, 101, 92, 133, 90, 160, 0, 1
     77, 89, 201
450 DATA 160, 208, 18, 138, 145, 89, 169
     , 32, 145, 91
460 DATA 165, 89, 133, 87, 165, 90, 133,
      88, 76, +87
470 DATA 232, 138, 41, 3, 197, 1, 208, 1
     89, 177, 87
480 DATA 170, 169, 32, 145, 87, 224, 4,
     240, 26, 138
490 DATA 10, 168, 162, 2, 56, 165, 87, 2
     49, +0, 133
500 DATA 87, 165, 88, 249, +1, 133, 88,
     202, 208, 238
510 DATA 76, +87, 96, XXX
```

## **Program 3:** Maze Generator For The 64

```
10 I=49152:IFPEEK(I+2)=216THENSYS49160:E
   ND
20 READ A:IF A=256 THEN SYS 49160:END
30 POKE I,A:I=I+1:GOTO 20
49152 DATA 1,0,216,255,255,255,40
49160 DATA 0,169,81,133,251,169,40
49168 DATA 133,253,169,4,133,252,133
49176 DATA 254,169,147,32,210,255,162
49184 DATA 0,160,0,169,160,145,253
49192 DATA 200,192,39,208,249,24,165
49200 DATA 253,105,40,133,253,144,2
49208 DATA 230,254,232,224,23,208,229
49216 DATA 160,0,169,4,145,251,169
49224 DATA 255,141,15,212,169,128,141
49232 DATA 18,212,173,27,212,41,3
49240 DATA 133,173,170,10,168,24,185
49248 DATA 0,192,101,251,133,170,185
49256 DATA 1,192,101,252,133,171,24
49264 DATA 185,0,192,101,170,133,253
49272 DATA 185,1,192,101,171,133,254
49280 DATA 160,0,177,253,201,160,208
49288 DATA 18,138,145,253,169,32,145
49296 DATA 170,165,253,133,251,165,254
49304 DATA 133,252,76,62,192,232,138
49312 DATA 41,3,197,173,208,189,177
49320 DATA 251,170,169,32,145,251,224
49328 DATA 4,240,26,138,10,168,162
49336 DATA 2,56,165,251,249,0,192
49344 DATA 133,251,165,252,249,1,192
49352 DATA 133,252,202,208,238,76,62
49360 DATA 192,169,1,160,0,153,0
49368 DATA 216,153,0,217,153,0,218
49376 DATA 153,0,219,200,208,241,96,256
```

## **Program 4:** Munchmaze For 4.0 PETs

```
5 PRINT"{CLR}PLEASE WAIT...."
10 I=12288
20 READ A:IF A=256 THEN SYS 12311
30 POKE I,A:I=I+1:GOTO 20
12288 DATA 20,4,10,0,88,178
12294 DATA 187,40,171,84,73,41
12300 DATA 58,158,49,48,52,56
```

*Write For FREE Catalog*

VISA

(602) 855-3357

Call your order in and get a FREE program

# NEW
# VIC SOFTWARE CBM 64

Call your order in and get a FREE program

*Write For FREE Catalog*

MasterCard

(602) 855-3357

CRICKET

PARATROOPER

MOW

ALIEN INVASION

SNAKE OUT

**CRICKET** - From the company that brought you Asteroidz, Munchman and a host of other blockbusters. We now present CRICKET. This is a challenging game with a cast of characters you will love and hate. All you have to do is get Cherp from one side of the road to the river and then across the river. Not so fast though. First you have to figure out how to dodge the traffic and get to the center. Then how are you going to get across the river? Look here comes a log — even a turtle. Hitch a ride across the river and jump from one to the other. Keep a sharp eye out for Ade the Gator. He loves to have crickets for lunch. How many times can you get across the road and river. You will have to work as fast as you can. Time limit and bonus. You will find this game addictive and challenging and it will entertain you with hours of fun and enjoyment. **$14.95**

**BUG BLAST** - If you think Centipede was fun — look out for BUG BLAST. A new and fast action arcade game with realistic smooth action, quality hi-res graphics and trouble. Its very calm as the first wave attacks. Only a few bugs to kill. Just shoot thru the cactus and wipe them out. After a few attacks you feel you have everything under control. Now the attacks really start. Those protection areas have to go. Blast away. Will they ever stop? OK — the BUGS got me this time. Now its my turn, Just one more time — BUG BLAST — Now its your turn to get even. **$14.95**

**BOMB'S AWAY** - Can you stop him? The crazy bomber drops the bombs from the top of the screen. You get 3 buckets to catch them. Before you know it bombs are falling so fast you wonder when he will stop. Just when you think you have him under control your bucket gets smaller. Is your hand quicker than your eye? **Special $9.95**

**PARATROOPER** - You are the only one left to stop them. The sky is full of enemy choppers. Paratroopers keep dropping into your area with non-stop barrage of enemy troops. They are out to destroy you. This new game is an unbeatable blend of arcade action and quick thinking strategy. You must make every shot count — don't be to fast on the trigger. Every time you hit a chopper or paratrooper you get extra points. Wait until you see the climax of this game — you won't believe it! This is a multiple skill level game with razor-sharp graphics and sound. **$19.95**

**MOW** - Get ready for the fast and furious action of the craziest mower you have ever seen. How much grass can you cut? Joystick moves your mower around as fast as you dare. Watch out for granny's dafodils and grandpa's radio antenna. **$14.95**

**COSMIC CRUZER** - Bring the coin-op game into your VIC. 3 Scenarios. Your Cruzer moves over a mountainous landscape & into a tunnel of surface - to - air missle, silos and ground - to - air weapons. If you can make it in and out of the tunnel you fly into the asteroid field. Drop bombs and fire missiles at the fuel dumps to keep your fuel supply up. If you are really good you can get to the base and try, to destroy it. We don't know of any one that has hit the base yet. Maybe you will be the 1st. Cosmic Cruzer is a fun filled magnificently rendered home video game that will last for months of challenge. Highly addicting. Hi-Res Graphics, Color & Sound. **SPECIAL PRICE - $14.95**

**SPACE PAK** - Can you survive? 3 space games with the sights and sounds of arcade games. The excitement builds as the action is un-ending. Blast away at everything in sight. The alien attacks will stop at nothing to destroy you. Prepare for battle, there is no escape, unless you can help. Can you survive? Hi-Res, Color, Graphic & Sound. Joystick or keyboard. **$19.95**

**ALIEN INVASION** - Invaders from space are attacking your home planet. Hurry and man your lasers and prepare your robot forces for the inevitable attack of the Alien Invaders. The excitment builds as you command a battery of missile bases in a bunker. Each invader has a laser aimed right at you. Will they ever stop. Only you can save the Galaxy. You can compete with 4 people in the solar system. There are 20 levels of play. If you destroy the Aliens in the correct order you will receive bonus points. Can you get the top score? **$14.95**

**TARGET COMMAND** - The whole West Coast is being bombarded and only you can save it. You are at the controls of the missile launcher and hold the destiny of our country in your hands. It takes a cool head, not hand and fast reflexes to zap those missiles right out of the air. Get ready to pulverize — atomize and vaporize them. Oh, my God, those warheads are heading right for our ammo dumps. They are everywhere. NO ONE CAN SAVE US — EXCEPT YOU. You must move your laser into position and fire as fast as you dare. Time limit with arcade style excitement. Protect your ammo at all costs. 10 levels of play. **$14.95**

**SNAKEOUT** - Slip your snake into position and score by chomping the blocks. Watch the way you slither because your escape routes get smaller. 2 Bonus games included. **$14.95**

**HEAD-ON** - Please do not buy this game if you are the type that says "I'll play it just one more time". Players have been known to start playing HEAD ON at 8:30 p.m. and at 2 a.m., wonder where the time went? Have you ever tried to explain to someone why you played a game for five and a half hours. We know of no remedy for the addiction to HEAD ON except to beat the VIC on level 9. No one has done it, YET, will you? We think not. Move your car as fast as you can dare around the tracks. You get 3 cars and MUST avoid the computer car. Points for the most dots covered. Bonus cars, nine levels of play. **$14.95**

**ComputerMat** •Box 1664 C • Lake Havasu City, Az. 86403 ***Add $1.00 For Shipping***

BUG BLAST

TARGET COMMAND


COSMIC CRUZER

SPACE PAK

HEAD ON

```
12306 DATA 0,0,0,234,234,169
12312 DATA 12,141,76,232,169,0
12318 DATA 141,96,10,141,97,10
12324 DATA 169,147,32,210,255,169
12330 DATA 44,141,153,51,169,52
12336 DATA 141,154,51,162,31,169
12342 DATA 129,32,144,51,169,0
12348 DATA 141,98,10,141,99,10
12354 DATA 133,143,24,165,142,105
12360 DATA 2,133,2,165,142,197
12366 DATA 2,208,250,169,147,32
12372 DATA 210,255,32,92,51,162
12378 DATA 39,169,160,157,39,128
12384 DATA 157,151,131,202,208,247
12390 DATA 169,80,133,84,169,128
12396 DATA 133,85,162,21,160,0
12402 DATA 169,160,145,84,160,38
12408 DATA 145,84,32,174,51,202
12414 DATA 208,240,169,43,141,153
12420 DATA 51,169,53,141,154,51
12426 DATA 162,166,169,128,32,144
12432 DATA 51,162,0,169,32,157
12438 DATA 0,11,157,0,12,157
12444 DATA 0,13,157,0,14,232
12450 DATA 208,241,169,81,133,84
12456 DATA 169,40,133,86,169,11
12462 DATA 133,85,133,87,162,0
12468 DATA 160,0,169,160,145,86
12474 DATA 200,192,39,208,249,24
12480 DATA 165,86,105,40,133,86
12486 DATA 144,2,230,87,232,224
12492 DATA 23,208,229,160,0,169
12498 DATA 4,145,84,32,41,210
12504 DATA 165,140,41,3,133,1
12510 DATA 170,10,168,24,185,255
12516 DATA 51,101,84,133,88,185
12522 DATA 0,52,101,85,133,89
12528 DATA 24,185,255,51,101,88
12534 DATA 133,86,185,0,52,101
12540 DATA 89,133,87,160,0,177
12546 DATA 86,201,160,208,18,138
12552 DATA 145,86,169,32,145,88
12558 DATA 165,86,133,84,165,87
12564 DATA 133,85,76,213,48,232
12570 DATA 138,41,3,197,1,208
12576 DATA 189,177,84,170,169,32
12582 DATA 145,84,224,4,240,26
12588 DATA 138,10,168,162,2,56
12594 DATA 165,84,249,255,51,133
12600 DATA 84,165,85,249,0,52
12606 DATA 133,85,202,208,238,76
12612 DATA 213,48,169,193,141,153
12618 DATA 51,169,53,141,154,51
12624 DATA 162,130,138,32,144,51
12630 DATA 32,228,255,208,251,32
12636 DATA 228,255,240,251,201,81
12642 DATA 208,13,169,147,32,210
12648 DATA 255,32,92,51,169,13
12654 DATA 76,210,255,201,49,48
12660 DATA 230,201,52,16,226,56
12666 DATA 233,48,133,2,169,147
12672 DATA 32,210,255,162,0,189
12678 DATA 0,11,157,0,128,189
12684 DATA 0,12,157,0,129,189
12690 DATA 0,13,157,0,130,189
12696 DATA 0,14,157,0,131,232
12702 DATA 208,229,32,92,51,24
12708 DATA 165,2,105,176,141,0
12714 DATA 128,141,38,128,24,165
12720 DATA 143,105,60,133,255,165
12726 DATA 143,197,255,208,250,169
12732 DATA 81,133,84,133,88,169
12738 DATA 128,133,85,133,89,169
12744 DATA 1,133,255,162,2,134
12750 DATA 1,160,0,169,102,145
12756 DATA 84,169,0,133,143,166
12762 DATA 1,138,10,168,24,185
12768 DATA 255,51,101,84,133,86
12774 DATA 185,0,52,101,85,133
12780 DATA 87,160,0,177,86,201
12786 DATA 160,208,9,202,138,41
12792 DATA 3,133,1,76,217,49
12798 DATA 201,81,208,3,76,184
12804 DATA 50,169,102,145,86,169
12810 DATA 58,145,84,165,86,133
12816 DATA 84,165,87,133,85,232
12822 DATA 138,41,3,133,1,165
12828 DATA 255,240,8,160,0,132
12834 DATA 255,169,81,145,88,162
12840 DATA 0,165,151,221,7,52
12846 DATA 240,8,232,224,4,208
12852 DATA 246,76,107,50,138,10
12858 DATA 168,24,185,255,51,101
12864 DATA 88,133,90,185,0,52
12870 DATA 101,89,133,91,160,0
12876 DATA 177,90,201,160,240,25
12882 DATA 201,58,208,3,32,186
12888 DATA 51,160,0,169,81,145
12894 DATA 90,169,32,145,88,165
12900 DATA 90,133,88,165,91,133
12906 DATA 89,165,143,197,2,208
12912 DATA 250,173,98,10,201,16
12918 DATA 208,61,173,99,10,201
12924 DATA 39,208,54,169,81,133
12930 DATA 84,169,128,133,85,162
12936 DATA 0,160,0,177,84,201
12942 DATA 58,208,7,32,186,51
12948 DATA 169,32,145,84,200,192
12954 DATA 37,208,238,32,174,51
12960 DATA 232,224,21,208,228,162
12966 DATA 0,189,17,54,240,6
12972 DATA 157,51,128,232,208,245
12978 DATA 76,51,51,76,213,49
12984 DATA 169,102,145,86,169,58
12990 DATA 145,84,165,88,133,84
12996 DATA 165,89,133,85,56,165
13002 DATA 84,233,41,133,86,165
13008 DATA 85,233,0,133,87,169
13014 DATA 240,133,2,169,255,133
13020 DATA 143,165,86,133,88,165
13026 DATA 87,133,89,169,0,133
13032 DATA 1,160,0,162,0,177
13038 DATA 88,221,11,52,240,5
13044 DATA 232,224,8,208,246,134
13050 DATA 254,56,169,7,229,254
13056 DATA 170,189,11,52,145,88
13062 DATA 200,192,3,208,224,24
13068 DATA 165,88,105,40,133,88
13074 DATA 144,2,230,89,230,1
13080 DATA 165,1,201,3,208,203
13086 DATA 165,143,208,252,198,2
13092 DATA 208,179,162,0,189,34
13098 DATA 52,240,6,157,55,128
13104 DATA 232,208,245,56,173,98
13110 DATA 10,237,96,10,141,100
13116 DATA 10,173,99,10,237,97
13122 DATA 10,13,100,10,240,17
13128 DATA 144,15,173,98,10,141
```

# Fantastic Sale!

## HES Special Programs For VIC-20 Computer

**Call Today!!** **Call Today!!** **Call Today!!**

### SYNTHESOUND Sale $19.95

**• Fantastic Sound & Music • Piano Keyboard**

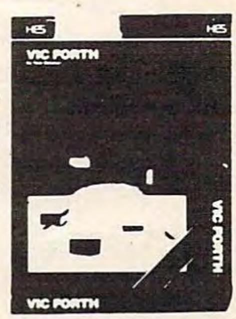

Now your VIC-20 can be used as a state-of-the-art music synthesizer and special effects generator! A very user friendly manual guides you through sessions of learning and development of your synthesizer "style"! You can program Synthesound to sound like many different instruments like bagpipes, organ, bass and many others. Synthesound brings technical musical abilities to any VIC-20 user. Prepare to step into the exciting world of the futuristic computer musician.

- VIC keyboard used as a 2 octave, fast attack keyboard.
- 3 voices with 4½ octave range.
- 4 low frequency ocillators.
- 4 envelope shaping controls (Attack, Decay, Sustain, Release).
- No additional hardware required.
- Spectacular special effects.
- 4 voice/20 note sequencer for repeating melodies or rhythms.
- High resolution piano keyboard display.

List $44.95 / **Sale $19.95**

### VIC FORTH LANGUAGE

**Faster than BASIC — easier than Assembly language — Language of Robots.**




VIC FORTH is an exciting new cartridge for your VIC-20 computer. You now have a language that is more powerful than BASIC and easier to program than assembler! VIC FORTH is a nearly complete implementation of the "FORTH INTEREST GROUP" (fig.) version of FORTH. The VIC FORTH editor follows closely the standard FORTH editor described in the book, "Starting FORTH", plus it has many, many additional features.

- Full screen editor which has 16 lines of 64 characters each (standard FORTH Screen) through use of a horizontal scrolling window.
- User definable words; this means a function not already supported can be created by you and added to VIC FORTH for future use.
- Full VIC sound and color capabilities.
- Runs in standard 5k VIC.
- Output to printers and disks and tape machines.
- Up to 24k additional memory can be added.
- 31 characters for variable names.

List $59.95 / **Sale $39.95**

### TURTLE GRAPHICS

**• Create colorful pictures and graphics • Learn how computers compute.**


Turtle Graphics is an exciting and fun way to discover computer programming concepts. Turtle Graphics is an easy-to-learn computer language. It enables the beginning programmer to master the computer in just a few minutes. By telling an imaginary turtle how to roam about on the VIC's screen, the Turtle Graphics programmer can create colorful and artistic pictures. As one becomes more and more adept at teaching the turtle to paint using the VIC's rich graphics and color palette he or she is painlessly learning all of the basics of computer programming. This not only develops computer awareness, but enriches one's math, logic, communication and artistic skills as well.

- Turtle Graphics has over 30 different commands.
- Programs may be listed on a printer and saved on or loaded from tape or disk.
- Turtle Graphics is menu driven for ease of use.
- Trace mode to help the beginning programmer follow the logic of his program one step at a time.
- Well written, easy-to-follow manual with many examples included.

List $39.95 / **Sale $29.95**

***We Have All HES Products At Super Sale Prices!!***

**PROTECTO ENTERPRIZES** (WE LOVE OUR CUSTOMERS)

**BOX 550, BARRINGTON, ILLINOIS 60010**
**Phone 312/382-5244 to order**

```
13134 DATA 96,10,173,99,10,141
13140 DATA 97,10,32,118,51,76
13146 DATA 58,48,162,0,189,19
13152 DATA 52,240,6,157,6,128
13158 DATA 232,208,245,162,0,189
13164 DATA 28,52,240,6,157,22
13170 DATA 128,232,208,245,172,96
13176 DATA 10,173,97,10,32,188
13182 DATA 196,32,147,207,162,0
13188 DATA 189,0,1,240,6,157
13194 DATA 27,128,232,208,245,96
13200 DATA 134,84,133,85,162,0
13206 DATA 160,0,189,44,52,240
13212 DATA 10,201,255,240,12,145
13218 DATA 84,200,232,208,241,32
13224 DATA 174,51,232,208,233,96
13230 DATA 24,165,84,105,40,133
13236 DATA 84,144,2,230,85,96
13242 DATA 138,72,152,72,24,173
13248 DATA 98,10,105,2,141,98
13254 DATA 10,144,3,238,99,10
13260 DATA 162,0,181,84,72,232
13266 DATA 224,8,208,248,172,98
13272 DATA 10,173,99,10,32,188
13278 DATA 196,32,147,207,162,0
13284 DATA 189,0,1,240,6,157
13290 DATA 12,128,232,208,245,162
13296 DATA 8,104,149,83,202,208
13302 DATA 250,104,168,104,170,96
13308 DATA 234,234,234,1,0,216
13314 DATA 255,255,255,40,0,54
13320 DATA 56,52,50,160,32,58
13326 DATA 102,170,186,127,255,19
13332 DATA 3,15,18,5,58,32
13338 DATA 48,0,8,9,7,8
13344 DATA 58,0,135,129,141,133
13350 DATA 160,143,150,133,146,0
13356 DATA 79,77,32,32,78,80
13362 DATA 99,80,32,79,99,79
13368 DATA 77,32,79,80,78,99
13374 DATA 99,99,77,79,80,32
13380 DATA 79,80,0,101,32,77
13386 DATA 78,32,103,32,103,32
13392 DATA 101,32,101,32,77,101
13398 DATA 103,32,32,79,76,100
13404 DATA 101,103,32,101,103,0
13410 DATA 101,32,32,32,32,103
13416 DATA 32,103,32,101,32,101
13422 DATA 32,32,32,103,32,32
13428 DATA 101,32,32,101,32,99
13434 DATA 32,103,0,101,103,77
13440 DATA 78,101,103,32,103,100
13446 DATA 101,32,101,103,77,32
13452 DATA 103,32,32,76,79,99
13458 DATA 101,103,99,101,103,0
13464 DATA 76,122,32,32,76,122
13470 DATA 77,100,100,100,78,76
13476 DATA 122,32,77,122,77,100
13482 DATA 100,100,78,76,122,32
13488 DATA 76,122,0,0,160,223
13494 DATA 32,32,233,231,32,233
13500 DATA 160,223,32,160,160,160
13506 DATA 160,231,160,160,160,160
13512 DATA 160,32,2,25,0,160
13518 DATA 160,223,233,160,231,233
13524 DATA 160,226,160,223,32,32
13530 DATA 233,160,105,160,160,0
13536 DATA 160,160,160,160,160,231
13542 DATA 160,160,98,160,231,32
13548 DATA 233,160,105,32,160,160
13554 DATA 160,160,32,32,7,1
13560 DATA 18,25,0,160,160,95
13566 DATA 105,160,231,160,160,226
13572 DATA 160,231,233,160,105,32
13578 DATA 32,160,160,0,160,160
13584 DATA 32,32,160,231,160,160
13590 DATA 32,160,231,160,160,160
13596 DATA 160,231,160,160,160,160
13602 DATA 160,32,13,1,18,19
13608 DATA 1,0,255,32,32,32
13614 DATA 184,32,32,32,32,32
13620 DATA 13,15,22,5,32,20
13626 DATA 8,5,32,34,81,34
13632 DATA 32,21,19,9,14,7
13638 DATA 0,32,32,32,30,32
13644 DATA 32,32,32,32,20,8
13650 DATA 5,32,14,21,13,2
13656 DATA 5,18,32,16,1,4
13662 DATA 46,0,32,32,32,93
13668 DATA 0,180,60,67,81,67
13674 DATA 62,182,32,32,184,32
13680 DATA 61,32,13,15,22,5
13686 DATA 32,21,16,0,32,32
13692 DATA 32,93,32,32,32,32
13698 DATA 32,180,32,61,32,13
13704 DATA 15,22,5,32,12,5
13710 DATA 6,20,0,32,32,32
13716 DATA 22,32,32,32,32,32
13722 DATA 182,32,61,32,13,15
13728 DATA 22,5,32,18,9,7
13734 DATA 8,20,0,32,32,32
13740 DATA 178,32,32,32,32,32
13746 DATA 178,32,61,32,13,15
13752 DATA 22,5,32,4,15,23
13758 DATA 14,0,255,3,8,15
13764 DATA 15,19,5,32,19,16
13770 DATA 5,5,4,32,6,1
13776 DATA 3,20,15,18,58,0
13782 DATA 0,32,32,32,177,32
13788 DATA 61,32,6,1,19,20
13794 DATA 44,32,178,32,61,32
13800 DATA 13,15,4,5,18,1
13806 DATA 20,5,44,32,179,32
13812 DATA 61,32,19,12,15,23
13818 DATA 0,0,15,18,32,16
13824 DATA 18,5,19,19,32,145
13830 DATA 32,20,15,32,17,21
13836 DATA 9,20,46,0,255,153
13842 DATA 143,149,167,146,133,160
13848 DATA 129,160,151,137,142,142
13854 DATA 133,146,161,161,0,256
```

## Program 5: Munchmaze For Upgrade PETs

*Use Program 4 but substitute these lines for Upgrade PET.*

```
12498 DATA 4,145,84,32,127,223
13176 DATA 10,173,97,10,32,109
13182 DATA 210,32,233,220,162,0
13272 DATA 10,173,99,10,32,109
13278 DATA 210,32,233,220,162,0
13314 DATA 255,255,255,40,0,41
13320 DATA 50,42,18,160,32,58
```

## Program 6: Munchmaze For The 64

```
10 I=12288 :POKE53281,1
20 READ A:IF A=256 THEN 35
30 POKE I,A:I=I+1:GOTO 20
35 SYS12311 :END
```

```
12288 DATA 20,4,10,0,88,178,187
12296 DATA 40,171,84,73,41,58,158
12304 DATA 49,48,52,56,0,0,0
12312 DATA 234,234,169,21,141,24,208
12320 DATA 169,0,141,96,10,141,97
12328 DATA 10,169,147,32,210,255,169
12336 DATA 7,162,0,157,0,216,157
12344 DATA 0,217,157,0,218,157,0
12352 DATA 219,232,208,241,169,120,141
12360 DATA 229,51,169,52,141,230,51
12368 DATA 162,31,169,5,32,220,51
12376 DATA 169,0,141,98,10,141,99
12384 DATA 10,133,162,24,165,161,105
12392 DATA 2,133,166,165,161,197,166
12400 DATA 208,250,169,147,32,210,255
12408 DATA 169,0,162,0,157,0,216
12416 DATA 157,0,217,157,0,218,157
12424 DATA 0,219,232,208,241,32,168
12432 DATA 51,162,39,169,160,157,39
12440 DATA 4,157,151,7,202,208,247
12448 DATA 169,80,133,168,169,4,133
12456 DATA 169,162,21,160,0,169,160
12464 DATA 145,168,160,38,145,168,32
12472 DATA 250,51,202,208,240,169,119
12480 DATA 141,229,51,169,53,141,230
12488 DATA 51,162,166,169,4,32,220
12496 DATA 51,162,0,169,32,157,0
12504 DATA 11,157,0,12,157,0,13
12512 DATA 157,0,14,232,208,241,169
12520 DATA 81,133,168,169,40,133,170
12528 DATA 169,11,133,169,133,171,162
12536 DATA 0,160,0,169,160,145,170
12544 DATA 200,192,39,208,249,24,165
12552 DATA 170,105,40,133,170,144,2
12560 DATA 230,171,232,224,23,208,229
12568 DATA 160,0,169,4,145,168,32
12576 DATA 151,224,165,143,41,3,133
12584 DATA 165,170,10,168,24,185,75
12592 DATA 52,101,168,133,180,185,76
12600 DATA 52,101,169,133,181,24,185
12608 DATA 75,52,101,180,133,170,185
12616 DATA 76,52,101,181,133,171,160
12624 DATA 0,177,170,201,160,208,18
12632 DATA 138,145,170,169,32,145,180
12640 DATA 165,170,133,168,165,171,133
12648 DATA 169,76,251,48,232,138,41
12656 DATA 3,197,165,208,189,177,168
12664 DATA 170,169,32,145,168,224,4
12672 DATA 240,26,138,10,168,162,2
12680 DATA 56,165,168,249,75,52,133
12688 DATA 168,165,169,249,76,52,133
12696 DATA 169,202,208,238,76,251,48
12704 DATA 169,13,141,229,51,169,54
12712 DATA 141,230,51,162,6,138,32
12720 DATA 220,51,32,228,255,208,251
12728 DATA 32,228,255,240,251,201,81
12736 DATA 208,32,169,147,32,210,255
12744 DATA 169,0,162,0,157,0,216
12752 DATA 157,0,217,157,0,218,157
12760 DATA 0,219,232,208,241,32,168
12768 DATA 51,169,13,76,210,255,201
12776 DATA 49,48,211,201,52,16,207
12784 DATA 56,233,48,133,166,169,147
12792 DATA 32,210,255,169,0,162,0
12800 DATA 157,0,216,157,0,217,157
12808 DATA 0,218,157,0,219,232,208
12816 DATA 241,162,0,189,0,11,157
12824 DATA 0,4,189,0,12,157,0
12832 DATA 5,189,0,13,157,0,6
12840 DATA 189,0,14,157,0,7,232
12848 DATA 208,229,32,168,51,24,165
12856 DATA 166,105,176,141,0,4,141
12864 DATA 38,4,24,165,162,105,60
12872 DATA 133,254,165,162,197,254,208
12880 DATA 250,169,81,133,168,133,180
12888 DATA 169,4,133,169,133,181,169
12896 DATA 1,133,254,162,2,134,165
12904 DATA 160,0,169,102,145,168,169
12912 DATA 0,133,162,166,165,138,10
12920 DATA 168,24,185,75,52,101,168
12928 DATA 133,170,185,76,52,101,169
12936 DATA 133,171,160,0,177,170,201
12944 DATA 160,208,9,202,138,41,3
12952 DATA 133,165,76,37,50,201,81
12960 DATA 208,3,76,4,51,169,102
12968 DATA 145,170,169,58,145,168,165
12976 DATA 170,133,168,165,171,133,169
12984 DATA 232,138,41,3,133,165,165
12992 DATA 254,240,8,160,0,132,254
13000 DATA 169,81,145,180,162,0,165
13008 DATA 197,221,83,52,240,8,232
13016 DATA 224,4,208,246,76,183,50
13024 DATA 138,10,168,24,185,75,52
13032 DATA 101,180,133,195,185,76,52
13040 DATA 101,181,133,196,160,0,177
13048 DATA 195,201,160,240,25,201,58
13056 DATA 208,3,32,6,52,160,0
13064 DATA 169,81,145,195,169,32,145
13072 DATA 180,165,195,133,180,165,196
13080 DATA 133,181,165,162,197,166,208
13088 DATA 250,173,98,10,201,16,208
13096 DATA 61,173,99,10,201,39,208
13104 DATA 54,169,81,133,168,169,4
13112 DATA 133,169,162,0,160,0,177
13120 DATA 168,201,58,208,7,32,6
13128 DATA 52,169,32,145,168,200,192
13136 DATA 37,208,238,32,250,51,232
13144 DATA 224,21,208,228,162,0,189
13152 DATA 93,54,240,6,157,51,4
13160 DATA 232,208,245,76,127,51,76
13168 DATA 33,50,169,102,145,170,169
13176 DATA 58,145,168,165,180,133,168
13184 DATA 165,181,133,169,56,165,168
13192 DATA 233,41,133,170,165,169,233
13200 DATA 0,133,171,169,240,133,166
13208 DATA 169,255,133,162,165,170,133
13216 DATA 180,165,171,133,181,169,0
13224 DATA 133,165,160,0,162,0,177
13232 DATA 180,221,87,52,240,5,232
13240 DATA 224,8,208,246,134,253,56
13248 DATA 169,7,229,253,170,189,87
13256 DATA 52,145,180,200,192,3,208
13264 DATA 224,24,165,180,105,40,133
13272 DATA 180,144,2,230,181,230,165
13280 DATA 165,165,201,3,208,203,165
13288 DATA 162,208,252,198,166,208,179
13296 DATA 162,0,189,110,52,240,6
13304 DATA 157,55,4,232,208,245,56
13312 DATA 173,98,10,237,96,10,141
13320 DATA 100,10,173,99,10,237,97
13328 DATA 10,13,100,10,240,17,144
13336 DATA 15,173,98,10,141,96,10
13344 DATA 173,99,10,141,97,10,32
13352 DATA 194,51,76,77,48,162,0
13360 DATA 189,95,52,240,6,157,6
13368 DATA 4,232,208,245,162,0,189
13376 DATA 104,52,240,6,157,22,4
13384 DATA 232,208,245,172,96,10,173
```

```
13392 DATA 97,10,32,145,179,32,221
13400 DATA 189,162,0,189,0,1,240
13408 DATA 6,157,27,4,232,208,245
13416 DATA 96,134,168,133,169,162,0
13424 DATA 160,0,189,13,54,240,10
13432 DATA 201,255,240,12,145,168,200
13440 DATA 232,208,241,32,250,51,232
13448 DATA 208,233,96,24,165,168,105
13456 DATA 40,133,168,144,2,230,169
13464 DATA 96,138,72,152,72,24,173
13472 DATA 98,10,105,2,141,98,10
13480 DATA 144,3,238,99,10,162,0
13488 DATA 181,168,72,232,224,8,208
13496 DATA 248,172,98,10,173,99,10
13504 DATA 32,145,179,32,221,189,162
13512 DATA 0,189,0,1,240,6,157
13520 DATA 12,4,232,208,245,162,8
13528 DATA 104,149,167,202,208,250,104
13536 DATA 168,104,170,96,234,234,234
13544 DATA 1,0,216,255,255,255,40
13552 DATA 0,37,33,34,36,160,32
13560 DATA 58,102,170,186,127,255,19
13568 DATA 3,15,18,5,58,32,48
13576 DATA 0,8,9,7,8,58,0
13584 DATA 135,129,141,133,160,143,150
13592 DATA 133,146,0,79,77,32,32
13600 DATA 78,80,99,80,32,79,99
13608 DATA 79,77,32,79,80,78,99
13616 DATA 99,99,77,79,80,32,79
13624 DATA 80,0,101,32,77,78,32
13632 DATA 103,32,103,32,101,32,101
13640 DATA 32,77,101,103,32,32,79
13648 DATA 76,100,101,103,32,101,103
13656 DATA 0,101,32,32,32,32,103
13664 DATA 32,103,32,101,32,101,32
13672 DATA 32,32,103,32,32,101,32
13680 DATA 32,101,32,99,32,103,0
13688 DATA 101,103,77,78,101,103,32
13696 DATA 103,100,101,32,101,103,77
13704 DATA 32,103,32,32,76,79,99
13712 DATA 101,103,99,101,103,0,76
13720 DATA 122,32,32,76,122,77,100
13728 DATA 100,100,78,76,122,32,77
13736 DATA 122,77,100,100,100,78,76
13744 DATA 122,32,76,122,0,0,160
13752 DATA 223,32?,32,233,231,32,233
13760 DATA 160,223,32,160,160,160,160
13768 DATA 231,160,160,160,160,160,32
13776 DATA 2,25,0,160,160,223,233
13784 DATA 160,231,233,160,226,160,223
13792 DATA 32,32,233,160,105,160,160
13800 DATA 0,160,160,160,160,160,231
13808 DATA 160,160,98,160,231,32,233
13816 DATA 160,105,32,160,160,160,160
13824 DATA 32,32,7,1,18,25,0
13832 DATA 160,160,95,105,160,231,160
13840 DATA 160,226,160,231,233,160,105
13848 DATA 32,32,160,160,0,160,160
13856 DATA 32;32,160,231,160,160,32
13864 DATA 160,231,160,160,160,160,231
13872 DATA 160,160,160,160,160,32,13
13880 DATA 1,18,19,1,0,255,32
13888 DATA 32,32,9,32,32,32,32
13896 DATA 32,13,15,22,5,32,20
13904 DATA 8,5,32,34,81,34,32
13912 DATA 21,19,9,14,7,0,32
13920 DATA 32,32,30,32,32,32,32
13928 DATA 32,20,8,5,32,12,5
13936 DATA 20,20,5,18,19,58,32
13944 DATA 32,32,0,32,32,32,93
13952 DATA 0,10,60,67,81,67,62
13960 DATA 11,32,32,9,32,61,32
13968 DATA 13,15,22,5,32,21,16
13976 DATA 0,32,32,32,93,32,32
13984 DATA 32,32,32,10,32,61,32
13992 DATA 13,15,22,5,32,12,5
14000 DATA 6,20,0,32,32,32,22
14008 DATA 32,32,32,32,32,11,32
14016 DATA 61,32,13,15,22,5,32
14024 DATA 18,9,7,8,20,0,32
14032 DATA 32,32,13,32,32,32,32
14040 DATA 32,13,32,61,32,13,15
14048 DATA 22,5,32,4,15,23,14
14056 DATA 0,255,160,32,160,32,32
14064 DATA 32,32,3,8,15,15,19
14072 DATA 5,32,19,16,5,5,4
14080 DATA 32,6,1,3,20,15,18
14088 DATA 58,0,0,160,32,160,32
14096 DATA 32,177,32,61,6,1,19
14104 DATA 20,44,32,178,32,61,32
14112 DATA 13,15,4,5,18,1,20
14120 DATA 5,44,32,179,32,61,32
14128 DATA 19,12,15,23,0,0,160
14136 DATA 32,160,32,32,32,32,32
14144 DATA 32,32,32,32,32,32,32
14152 DATA 32,16,18,5,19,19,32
14160 DATA 145,32,20,15,32,17,21
14168 DATA 9,20,46,0,255,153,143
14176 DATA 149,167,146,133,160,129,137
14184 DATA 142,142,133,146,161,161,32
14192 DATA 255,32,256
```

©

**D.E.S.-SOFT ™**
a division of
DES-Data Equipment Supply Corp.

# ZARCON

By Steven Prentiss

Destined to be one of the best arcade style games on commodore computers.Can you in your Zarcon Fighter shoot all the alien formations while flying through an asteriod field!

5 Levels of excitement
Joystick required
Vic-20 cassette 9.95
Commodore 64 cassette 9.95

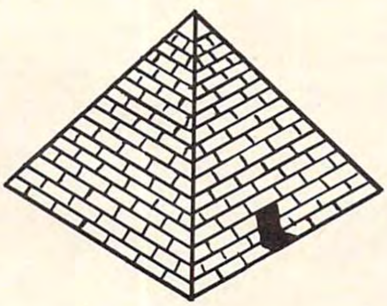
---

# SHROOMS

Subterranean Encounter
By Michael Koberstein

Inner Earth has invaded the surface world. The mutant spiders and the moles have taken humans captive and changed them into mussrooms. To rescue them you must enter the caverns. You are their only hope. Good Luck!
Vic-20 & 8K Expander,-Joystick-cass 9.95


---



# SPHINX

3-D ADVENTURE

By Cory Christensen

You the Adventurous Explorer are looking for the lost treasure of the Sphinx. It is rumored to be in the Ancient Pyramid of RA. Beware of deadly traps and hidden passage ways.
Commodore 64-cassette 9.95

---

You are the commander of squadron of laser ships. It is your duty to defend the cities of earth against incoming alien attack. Spectacular Hi-Res graphics and machine code for super fast action.
Joystick Vic-20 cassette 9.95


## SEE YOUR LOCAL DEALER TO SEE OUR FINE PRODUCTS

**Dealer inquiries invited** *Software Distribution Available* **Programs wanted**

**(714) 778-5455** D E S **Data Equipment Supply Corp. 8315 Firestone Blvd., Downey, CA 90241** **(213) 923-9361**

# ATARI®  Printers/Etc.

**600XL** .................. CALL
**1200XL** .................. CALL*
*Reflects $100 Atari Rebate

| | |
|---|---|
| 1050 DRIVE | $335 |
| 1025 PRINTER | $399 |
| 1020 COLOR PTR | $219 |
| 1027 PRINTER | $279 |
| 1010 RECORDER | $72 |
| 850 INTERFACE | CALL |
| MICROBITS INFC | $78 |
| 80 COLUMN BD | $249 |
| TECHNICAL NOTES | $25 |
| REAL TIME CLK | $38 |
| 810 DRIVE | $419 |

## DRIVES

| | |
|---|---|
| RANA 1000 | $319 |
| PERCOM 88-S1 | $358 |
| PERCOM AT 88-S2 | $539 |
| PERCOM 40-S1 | $510 |
| PERCOM 40-S2 | $799 |
| PERCOM 44-S1 | $639 |
| PERCOM 44-S2 | $929 |

## MEMORIES

| | |
|---|---|
| 48K RAM (INTEC) | $95 |
| 64K RAM (INTEC) | $119 |
| 48K RAM (MOSAIC) | $109 |
| 64K RAM (MOSAIC) | $145 |
| 128K RAM DISK | $299 |
| 32K RAM (MOSAIC) | $68 |

**GEMINI 10X** .. $289
**GORILLA** .... $199
**PROWRITER** . $345
**SMITH TPI** ... $488

**CITOH**
Prowriter .......... $345
Prowriter II ........ $629
Starwriter ........ $1149
Printmaster ....... $1448

**NEC**
8023 A-C .......... $409
3510 ............. $1375
3530 ............. $1579
3550 ............. $1779
7710/7730 ........ $1998

**SILVER REED P** ...... $669
**QUME 11/40 +** .... $1299

**OKI-DATA**
Microline 82A ...... $398
Microline 83A ...... $638
Microline 84P ...... $958
Microline 92 ....... $488
Microline 93 ....... $858

**DIABLO**
620R ............. $939
630R ............. $1719

## MONITORS

**AMDEK**
Color I .......... $289
V300 ............. $139
V300A ............ $149
Color II ......... $449

**NEC**
GRN (JB1260) ...... $115
GRN (JB1201) ...... $155
Color Composite .... $298
RGB Color ......... $598

## MODEMS

**HAYES**
Smartmodem ....... $209
Smartmodem 1200 .. $498
Micromodem II ..... $259

**NOVATION**
J-Cat ............. $99
Apple Cat II ....... $259
D-Cat ............ $149

**ANCHOR AUTOMATION**
Mark I or II Modem ... $78

# *ASTRA 1620
DISK DRIVE SYSTEM

**MORE FOR YOUR MONEY
DOUBLE OR SINGLE DENSITY
TWO DRIVES**

## SPECIAL ............ $469

## ATARI SOFTWARE

D = DISK
R = ROM (Cartridge)
C = CASSETTE

**ADVENTURE INT'L**
Adv. 1-12 each (C) .... $18
Preppie (C/D) ........ $20
Preppie II (C/D) ...... $23
Diskey (D) ........... $33
Sea Dragon (C/D) .... $23
Bug Off! (C/D) ....... $20
Tutti-Frutti (C/D) .... $17
Saga 1-3 (ea.) (D) .... $27
Lunar Lndr . (C) $11 (D) $15
Galactic Empire (C) .. $14
Galactic Trader (C) ... $14
Stratos (C/D) ........ $23
Rear Guard (C) $14 (D) $17

**APX**
Eastern Front (C/D) .. $23
747 Land Sim. (C/D) .. $17
Fig-Forth (C) ........ $30
Family Cash Flow (D) . $17
Downhill (C/D) ....... $17
Avalanche (C/D) ...... $17
Outlaw/Howitzer (C/D) $17
Salmon Run (C/D) .... $17
Data Management (D) $17
Galahad/Holy Grail (D) $21
Adv. Music System (D) $21

**ATARI INC.**
Microsoft Basic II (R) . $62
Mickey in Great Outdoors (C/D) .... $36
Paint (D) ............ $30
Speed Reading (C) ... $54
Qix (R) .............. $30
Dig Dug (R) .......... $30
Atari Writer (R) ...... $68
Time Wise (D) ....... $23
Visicalc (D) ........ $139
Juggles House (C/D) . $22
Juggles Rnbw (C/D) .. $22
Pilot (Home) (R) ...... $55
Galaxian (R) ......... $30
Defender (R) ......... $30
ET .................. $34
Microsoft Basic (D) .. $62
Macro Ass. & Edit (D) . $62
Assembler Editor (R) . $42
Basic Cartridge (R) .. $45
Pac Man (R) ......... $30
Centipede (R) ........ $30
Caverns of Mars (D) .. $28
Star Raiders (R) ...... $30
Conv. Lang. Ea. (C) ... $42
Music Composer (R) .. $31
Super Breakout (R) ... $26
My First Alphabet (D) . $26
Prog.2 & E (ea.)(C) ... $21
Word Processor (D) . $102
Pilot (Educ.) (R) ...... $92
Touch Typing (C) ..... $19
Home File Mngr (D) ... $36
Bookkeeper (D) ...... $102
Family Finance (D) ... $35
Prog. 1 (C) .......... $18
Scram (C) ........... $18
Asteroids (R) ........ $27
Space Invaders (R) ... $27
Missile Command (R) . $27
Telelink (R) .......... $21
Superman III (R) ..... $34
Basketball (R) ....... $24
Donkey Kong (R) ..... $34

**AVALON HILL**
VC (D) ............... $17
B-1 Nuc. Bomber (C) .. $12
Legionnaire (C) ...... $23
Empire of Overmind ..... (D) $23 (C) $20
Tanktics ... (D) $20 (C) $17
Comptr Stock & Bonds ..... (D) $17 (C) $14
Dnieper River Line (D) $20
Voyager (D) ......... $17
Controller (D) ....... $20
GFS Sorceress ......... (D) $23 (C) $20
Telegard .. (D) $19 (C) $16
Vorrak (D) ........... $18
Lord of Karma (C) .... $14

**AVALON HILL** (Cont'd)
Moon Patrol (C) ...... $17
Planet Miners (C) ..... $12
Conflict 2500 (C) ..... $12
Space Station Zulu ......... (D) $17 (C) $14
Flying Ace . (D) $21 (C) $18
Gypsy ..... (D) $18 (C) $15
Galaxy .... (D) $17 (C) $14
Guns of Fort Def. (C) . $14
Nuke War (C) ........ $12
Andromeda Conquest ......... (D) $16 (C) $13
Close Asslt (D) $23 (C) $20

**BRODERBUND**
Sky Blazer (D) ....... $22
Bank St. Writer (D) ... $46
A.E. (D) ............. $23
Arcade Machine (D) .. $39
Choplifter . (D) $23 (R) $29
Labyrinth (C/D) ...... $20
Serpentine . (D) $23 (R) $27
Steller Shuttle (C/D) .. $17
Apple Panic (C/D) .... $20
Genetic Drift (C/D) ... $20
David's Midnight (D) .. $23
Sea Fox (D) ......... $20
Track Attack (D) ..... $20
Operatn Whrlwnd (D) . $23
Match Boxes (C/D) ... $20

**CDY CONSULTING**
Pogoman (C/D) ...... $27
Mad-Netter (C/D) ..... $23
It-Is-Balloon (C/D) .... $23

**CBS**
Mountain King (R) ... $27
Boulders & Bombs (R) $27
Krazy (each) (R) ...... $27

**CONTINENTAL SOFT.**
Home Accountant (D) $48
Tax Advantage (D) ... $39

**DATA MOST**
Pig Pen (D) .......... $20
Mating Zone (D) ..... $20
Roundabout (D) ...... $20
Bilestoad (D) ........ $20
Night Raiders (D) .... $20
Monster Smash (D) ... $20

**DATASOFT**
Text Wizard (D) ...... $65
Graphic Master (D) ... $27
Micro Painter (D) .... $23
Lisp Interpreter (D) .. $79
Graphics Gen. (D) .... $17
Basic Compiler (D) ... $65
Zaxxon (C/D) ........ $27
Teletalk (D) ......... $34
Pac. Coast Hwy (C/D) . $20
Clowns/Balloons (C/D) $20
Spell Wizard (D) ..... $53
Canyon Climber (C/D) $20
Sands of Egypt (D) ... $27
O'Riley's Mine (C/D) .. $23
Rosen's Brigade (C/D) $23
Moon Shuttle (D) ..... $27

**DON'T ASK**
Sam (D) ............. $39
P.M. Animator (D) .... $23
Teletari (D) .......... $27
Poker Sam (D) ....... $17
Word Race . (D) $17 (C) $14

**EDU-WARE**
Prisoner II (D) ....... $27
Spelling Bee (D) ..... $27
Rendevous (D) ....... $27
Compu/Read ......... (D) $21 (C) $15
Compu/Math-FR ......... (D) $27 (C) $21
Compu/Math-Dec ......... (D) $27 (C) $21

**EDUCATIONAL SOFT.**
Tricky Tutorial 1,2,3 or 4 (C/D) ..... $15
Tricky Tutorial 5,6 or 7 (C/D) ...... $22

**EPYX-AUTO. SIMULATION**
Rescue at Rigel (C/D) . $20
Datestones/Ryn (C/D) $14
Upr Rchs Apshai (C/D) $14
Ricochet (C/D) ....... $14
Keys/Acheron (C/D) .. $14
Curse of Ra (C/D) .... $14
Danger/Drindisti (C/D) $14
Jumpman (C/D) ...... $27
Escp Vulcans Isle (D) . $20
Crypt of Undead (D) .. $20
Nightmare (D) ........ $20
Armor Assault (D) .... $27
Monster Maze (R) .... $27
Alien Garden (R) ..... $27
Plattermarna (R) ..... $27
Morloc's Tower (C) ... $14
King Arthur's Heir ......... (D) $20 (C) $14
Hellfire Warrior (C/D) . $27
Invasion Orion (C/D) .. $17
Temple of Aps. (C/D) . $27
Star Warrior (C/D) .... $27
Dragon's Eye (D) ..... $20
Crush Crumble (C/D) . $20

**GEBELLI**
Firebird (R) ......... $27
Embargo (R) ......... $30
Candy Factory (D) .... $20
Match Racers (C/D) .. $20
Dr. Goodcode (D) .... $20

**HAYDEN**
Go ........ (D) $23 (C) $20
Sargon II .. (D) $23 (C) $20
Bulldog Pinball (C) ... $20

**HES**
Coco (C/D) .......... $34
Coco II (C/D) ........ $27

**INFOCOM**
Suspended (D) ....... $34
Zork I, II or III (D) ..... $27
Starcross (D) ........ $27
Deadline (D) ......... $34
Witness (D) .......... $34

**INHOME**
Baseball .. (R) $23 (D) $23
Crypts of Terror ......... (D) $23 (C) $20

**INNOVATIVE DESIGN**
Pool 1.5 (D) .......... $23
Juggler (D) .......... $20
Speedway Blast ......... (D) $20 (R) $27
Pool 400 (R) ......... $27

**JV SOFTWARE**
Jrny to Plnts (C/D) .... $20
Action Quest (C/D) ... $20
Ghost Encount. (C/D) . $20

**LJK**
Letter Perfect (D) ... $104
Data Perfect (D) ..... $74
Letter Perfect (R) ... $137
Edit 6502 (R) ....... $137

**MONARCH DATA SYS.**
ABC Compiler (D) .... $48

**ODESTA**
Chess (D) ........... $45
Checkers (D) ........ $34
Odin (D) ............ $34

**OPTIMIZED SYSTEMS**
C-65 (D) ............ $58
Bug-65 (D) .......... $23
Max-65 (D) .......... $58
Basic A + (D) ........ $58
Action (D) ........... $65
Speed Read Plus (D) .. $40

**PHOENIX**
Birth of Phoenix (D) .. $16
Adv. In Time (D) ..... $20

**QUALITY**
Fastgammon (C) ..... $14
Name That Song ......... (D) $13 (C) $11
Starbase Hyperion ......... (D) $17 (C) $14
Ali Baba/40 Thvs (D) .. $20
Jeeper Creepers (D) .. $20

**ROKLAN**
Gorf ...... (D) $27 (R) $30
Wizard/Wor . (D) $27 (R) $30
Dlx Invaders (D) $23 (R) $27
Anti Sub Ptrl (D) $20 (C) $14
Telecom (D) ......... $49
Space Journey (R) ... $30

**SENTIENT**
Gold Rush (D) ....... $23
Cyborg (D) .......... $23

**SIERRA ON-LINE**
Mission Asteroids (D) $17
Ulys.& Gldn Fleece (D) $27
Crossfire .. (D) $20 (R) $23
Mouseattack (D) ..... $23
Jawbreaker (D) $20 (R) $23
Threshold (D) ....... $27
Softporn (D) ......... $20
Ultima II (D) ......... $39

**SIERRA ON-LINE**
Marauder (D) ........ $23
Lunar Leeper (D) .... $20
Wiz & Princess (D) ... $22
Frogger (C/D) ........ $23

**SIRIUS**
Alpha Shield (R) ..... $27
Wavy Navy (D) ....... $23
Bandits (D) .......... $23
Space Eggs (D) ...... $20
Sneakers (D) ........ $20
Way Out (D) ......... $27
Type Attack (D) ...... $27
Repton (D) ........... $27
Critical Mass (D) ..... $27
Fast Eddy (R) ........ $23
Worm War (R) ........ $23

**SPINNAKER**
Snooper Troop 1,2 (D) . $30
Kindercomp (D) ...... $20
Rhymes & Riddles (D) $20
Hey Diddle Diddle (D) $20
Srch Amzng Thngs (D) $27
Story Machine (D) .... $23
Face Maker (D) ...... $23

**STRATEGIC SIM.**
Cosmic Balance (D) .. $27
Cosmic Balance II (D) $27
Tigers In Snow (C/D) . $27
Battle of Shiloh (C/D) $27
Battle of Norm. (C/D) . $27
Galactic Gladiator (D) $27
Cytron Masters (D) ... $27
Knights/Desert (C/D) . $27
Combat Leader (C/D) . $27

**SUBLOGIC**
Pinball (C/D) ........ $20

**SWIFTY**
J. White Music Lessons ......... (C/D) $20
Space Shuttle (D) .... $20

**SYNAPSE SOFTWARE**
File Mngr 800 + ..... $65
Protector II (D) $23 (R) $29
Shamus ... (D) $23 (R) $29
Fort Apocalypse (C/D) $23
Shamus II (C/D) ...... $23
Necromancer (C/D) ... $23
Pharoh's Curse (C/D) . $23
Slime (C/D) .......... $23
Page 6 (D) .......... $23
Reptilian (C/D) ....... $23
Picnic Paranoia (C/D) . $23
Claim Jumper (C/D) .. $23
Drelbs (C/D) ......... $23
Shadow World (C/D) .. $23
Survivor (C/D) ........ $23
T.N.T. (C/D) .......... $23
N.Y.C. (C/D) .......... $23
Slamball (C/D) ....... $23
Blue Max (C/D) ....... $23
Nautilus (C/D) ....... $23
Chicken (R/D) ........ $23
Quasimodo (C/D) .... $23
River Quest (C/D) .... $23

**SYNERGISTIC**
Programr Wkshp (D) .. $23
Graph Workshop (D) .. $27
Disk Workshop (D) ... $23
Worlock's Revenge (D) $23

**THORN EMI**
Soccer (R) .......... $34
Jumbo Jet (R) ........ $34
Submarine Comm. (R) $34
Kick Back (R) ........ $34
Pool (C) ............. $22
Humpty Dumpty (C) .. $22
Hckry Dckry Dck (C) .. $22
Darts (C) ............ $22

**USA**
Atari World (D) ...... $39
3-D Sprgrphcs (C/D) .. $27
Survival Adv. (C/D) ... $17

**VERSA**
Mind Bgglrs I (D) $14 (C) $12
Globe Master (D) ..... $20

**MISCELLANEOUS**
Financial Wizard (D) .. $41
Castle Wolfenstein (D) $20
Master Type (D) ..... $27
Astro Chase (D) ..... $22
Miner 2049er (R) ..... $34
Cypher Bowel (C) .... $33
Bug Attack (C/D) ..... $20
Raster Blaster (D) ... $20
Air Strike (C/D) ...... $27
Prism (D) ........... $18
Megalegs (C) ........ $23
Cap'n Cosmo (D) ..... $19
Spy's Demise (D) ..... $14
Galac. Chase (D) $20 (C) $18
Kid Grid (C/D) ........ $20
Battle Trek (D) ....... $20
Snapper (D) ......... $19
T&F-PMP Property Management (D) .. $149
Millionaire (D) ....... $39

COSMIC™

## SPECIALS

| | |
|---|---|
| Gemini 15X Printer | $379 |
| Axiom AT-100 Printer (with interface | $229 |
| Astra Double Density Dual Drive | $469 |
| Rana 1000 Drive | $319 |
| Bit-3 80 Column Board | $245 |
| Mannesmann Talley 160L Printer | $589 |
| Atari 400 Keyboard (In Home) | $35 |
| Programmer Kit .. $48 | Entertainer Kit ... $64 |
| Wico Joystick ... $23 | Wico Trackball ... $49 |

# COSMIC COMPUTERS
UNLIMITED

**727 BREA CANYON RD., SUITE 16
WALNUT, CA 91789**

**ORDER LINES OPEN MON-SAT 8 am - 8 pm**

# (800) 626-7642
**PLEASE FOR ORDERS ONLY
SORRY, NO COD'S**

# (714) 594-5204
**FOR TECHNICAL INFO, ORDER INQUIRIES, OR FOR CALIFORNIA ORDERS**

Add $2.50 shipping per software order in continental U.S. Add $5.00 shipping per software order for AK, HI, FPO-APO. Add $10.00 or 15% (whichever is greater) per software order for non-U.S. Call for cost of hardware shipping. Calif. residents add 6½% sales tax. Cashiers checks or money orders filled within 24 hours for items in stock. Personal checks require 4 weeks to clear. MasterCard and Visa OK for software only within continental U.S., add 3% surcharge. Include card no., expiration date and signature. Due to our low prices, all sales are final. All defective returns must have a return authorization number. Please call to obtain one before returning goods for replacement or repair. Prices & availability subject to change.

# VIC 20

## 1541 DISK DRIVE ......... $239

| | | | |
|---|---|---|---|
| 1701 Color Monitor | $255 | 1530 Recorder | $59 |
| 1525 Printer | $239 | 1600 Modem | $59 |
| 1520 Color Ptr | $169 | 1650 Auto Modem | $158 |
| Hescard | $23 | Microtek 16K Ram | $82 |
| Hes Sound Box | $13 | UMI 3K Ram | $56 |
| Data 20 Exp. Chassis | $56 | | |

***Large Selection of Vic 20 Software at Great Prices!***

## VIC 20 SOFTWARE VIC 20

**APPLIED SYSTEMS**
Number Gulper (C) ... $17
Number Chaser (C) ... $17

**BRODERBUND**
Martian Raider (C) ... $14
Multisound Synth. (C) $14
Shark Trap (C) ... $14
Sky Blazer (R) ... $27
Seafox (R) ... $27
AE (R) ... $27

**CBS**
K-Razy Patrol (R) ... $29
K-Razy Antiks (R) ... $29

**COMM-DATA**
Pakacuda (C) ... $13
Sketch and Paint (C) .. $13
Eight Ball (C) ... $13
Invader Educ. Series (each) (C) ... $13

**COMPUTERMAT**
Paratroopers (C) ... $15
Bug Blast (C) ... $12
Cricket (C) ... $12

**CREATIVE S.W.**
Black Hole (R) ... $36
Trashman (R) ... $36
Astroblitz (R) ... $36
City Bomber (R) ... $20
Apple Panic (R) ... $36
Choplifter (R) ... $36
Serpentine (R) ... $36
Videomania (R) ... $36
Terraguard (R) ... $36

**EPYX**
Monster Maze (R) ... $27
Sword of Fargoal (C) . $20
Ricochet (C) ... $14
Rescue at Rigel (C) ... $20
Temple of Apshai (D) . $27

**HES**
Gridrunner (R) ... $27
Raid of Isram (C) ... $14
Robot Panic (R) ... $27
Protector (R) ... $29
Shamus (R) ... $27
Predator (R) ... $27
Syn The Sound (R) ... $39
Aggressor (R) ... $27
Heswriter (R) ... $27
Turtle Graphics (R) ... $27
Hesmon (R) ... $27
Vic Forth (R) ... $39
6502 Prof.Dev.Sys. (C) $20
Torg (C) ... $13
Concentration (C) ... $12
Fuel Pirates (C) ... $12
Simon (C) ... $12
Vic Trek (C) ... $13
Co Co II (C/D) ... $27

**IMAGIC**
Demon Attack (R) ... $25
Atlantis (R) ... $25

**LITTLE WIZARD**
Pro Football (C) ... $15
Cosmic Crusader (C) . $12

**LOGISTIC**
X-Rated (C) ... $9
Algebra Wiz (C) ... $9
Profess. Journal (C) .. $36
Viccalc (C) ... $36
Budget & Bill Payer (C) $15
Accounts Payable (C) $36
Mad Bomber (C) ... $10

**MIS**
Vic Vango (C) ... $10
Galactic Crossfire (C) $11
Checkbook (C) ... $14
Fruitfly (C) ... $10
Alien Soccer (C) ... $11

**NUFEKOP**
Krazy Kong (C) ... $11
3-D Man (C) ... $17
Exterminator (C) ... $20

**ON LINE**
Crossfire (C) ... $20
Ultima (C) ... $14

**QUICK BROWN FOX**
Prof. Word. Proc. ... $48

**SIRIUS**
Snake Byte (R) ... $27
Fly Wars (R) ... $27
Type Attack (R) ... $27
Fast Eddy (R) ... $27
Deadly Duck (R) ... $27
Turmoil (R) ... $27
Plasmania (R) ... $27
Spider City (R) ... $27
Squish 'Em (R) ... $27
Final Orbit (R) ... $27
Bumper Bash (R) ... $27

**SPECTRAVISION**
Cave In (R) ... $27
Number Crunch (R) ... $27
Reaganomics (R) ... $27
Gold Mine (D) ... $27
Ape Escape (D) ... $27

**STARTECH**
Asteroids (C) ... $18
Backgammon (C) ... $18
Ski Run (C) ... $12
Meteor (C) ... $9
Vic Men (C) ... $18
Vic Panic (C) ... $18

**SYNAPSE**
Harrier (C) ... $23
Squeeze (C) ... $23
Astro Patrol (C) ... $23

**TAYLORMADE**
Fun Fractions (C) ... $18
Vic Lemonade (C) ... $12
Tch Typing Tutor (C) .. $13

**T&F SOFTWARE**
Word Search (C) ... $14
Sports Search (C) ... $14
Arcade Search (C) ... $14

**TOTL**
Mailing List (C) ... $14
Time Management (C) $22
Research Asst. (C) ... $22

**THORN EMI**
River Rescue (R) ... $29
Vic Music Comp. (R) .. $29
Submarine Comm. (R) $29
Mutant Herd (R) ... $29
Fourth Encounter (R) . $29

**TRONIX**
Galactic Blitz (C) ... $17
Swarm (C) ... $20
Sidewinder (C) ... $20
Scorpion (R) ... $27
Gold Fever (R) ... $27
Deadly Skies (R) ... $27

**UMI**
Video Vermin (R) ... $27
Amok (C/R) ... $20
Outworld (R) ... $27
Satellites & Met. (R) .. $27
Subchase (C) ... $17
Kosmic Kamikaze (C) . $17
Meteor Shower (C) ... $11
Super Hangman (C) .. $14
Spiders of Mars (R) ... $27
Meteor Run (R) ... $27
Vicalc (C) ... $11
Vi Term A (C) ... $14
Vi Cat (C) ... $17
Vi Check (C) ... $17
Alien Blitz (C/R) ... $20
Sky Math (C) ... $11
Space Division (C) ... $11
The Alien (C) ... $17
Grand Master (C) ... $27
Renaissance (R) ... $33
Cloud Burst (R) ... $20
Skibbereen (R) ... $20
Wordcraft 20 (R) ... $65

**VICTORY**
Adv. Pak I (C) ... $14
Adv. Pak II (C) ... $14
Annihilator (C) ... $18
Grave Robbers (C) ... $12
Kongo Kong (C) ... $18
Trek (C) ... $12

# 64 LATE ARRIVALS 64

**BATTERIES INCLUDED**
Delphis Oracle(D) ... 114
Paper Clip(D) ... 89

**COMM—DATA (New Versions)**
Supercuda(C/D) ... 18
Pegassus Odyssey(C/P) 18
Escape MCP(C/D) ... 18
Toddler Tutor(C/D) ... 18
Prim. Math Tutor(C/D) . 18
Math Tutor(C/D) ... 18
English Invaders(C/D) . 18

**DATA 20**
Pro Word Proc.(C) ... 21
General Ledger(C) ... 21
Accounts Rec.(C) ... 21
Elec. Spreedsheet(C) .. 21

**EN—TECH**
Studio 64(D) ... 28
Sprite Fun(C) ... 18

**EPYX**
sword Fargoal(C/D) ... 20
Crush Crumble(D) ... 20
Upper Reaches APS(D) 14

**HES**
Synthe Sound 64(R) ... 34
64 Forth(R) ... 40
Time/Money Mgr.(D) .. 48

**INFO—DESIGNS**
G/L (D) ... 61
A/P (D) ... 61
A/R (D) ... 61

**NUFEKOP**
Exterminator(C) ... 20
3-D Man(C) ... 16

**RAINBOW**
Pers. Finance Assist.(D) 41

**PSYCOM SOFTWARE**
Personality Analyzer(D) 24

**SIERRA ON—LINE**
Crossfire(D) ... 20

**SOUTHERN SOLUTIONS**
Bill Payer(A/P) ... 68
Business Man(G/L) ... 68
Paymaster(Payroll) ... 68
Bill Collector(A/R) ... 68
Widget(Inventory) ... 68

**T & F SOFTWARE**
Word Search(C) ... 15
Sport Search(C) ... 15
Arcade Search(C) ... 15

**TIMEWORKS**
Programming Kit 1(D) . 18
Programming Kit 2(D) . 18
Programming Kit 3 (D) . 18

D = Disk
R = ROM (Cartridge)
C = Cassette

COSMIC™

## SPECIALS

| | |
|---|---|
| Gemini 10X Printer | $289 |
| Prowriter Printer | $345 |
| Gorilla Banana Printer | $199 |
| Card ? Printer Interface | $60 |
| The Connection Printer Interface | $85 |
| 1541 Disk Drive | $239 |
| Koala Graphics Tablet | $69 |
| Flip N' File Diskette Box | $21 |
| Elephant Disks (10) | $21 |
| The Boss Joystick (Wico) | $15 |
| Wico Joystick | $23 |
| Wico Trackball | $49 |

# COSMIC COMPUTERS
UNLIMITED

**727 BREA CANYON RD., SUITE 16**
**WALNUT, CA 91789**
**ORDER LINES OPEN MON-SAT 8 am - 8 pm**

**(800) 626-7642**
PLEASE FOR ORDERS ONLY
SORRY, NO COD'S

**(714) 594-5204**
FOR TECHNICAL INFO, ORDER INQUIRIES, OR FOR CALIFORNIA ORDERS

# COMMODORE

## CBM 64 ......... CALL
## 1541 DISK DRIVE ... $239

| | | | |
|---|---|---|---|
| 1701 Color Monitor | $255 | 1530 Recorder | $59 |
| 1525 Printer | $239 | 1600 Modem | $59 |
| 1520 Color Ptr | $169 | 1650 Auto Modem | $158 |
| Card ? (Infc) | $60 | CMB 64 Ref Guide | $18 |
| Light Pen | $29 | The Connection (Infc) | $85 |
| Cassette Infc | $29 | MSD Disk Drive | $339 |
| Card ? Software | $16 | PTI 45 Lot Board | $59 |

***Large Selection of 64 Software at Great Prices***

## 64 SOFTWARE 64

**ACCESS SOFTWARE**
Neutral Zone (C/D) ... $26
Sprite Master (C/D) ... $27

**AVALON HILL**
Nukewar (C) ... $12
Planet Miners (C) ... $12
Androm. Conquest (C) $14
Midway Campaign (C) $12
North Atl. Convoy (C) $12
Comp. Stcks/Bnds (C) $15
Computer Football (C) $18
Telengard (C) ... $16

**BATTERIES INCLUDED**
Paper Clip (D) ... $89
D.Base ... $114

**BRODERBUND**
Choplifter (R) ... $29
Serpentine (R) ... $27
Seafox (R) ... $27
David's Midnight (D) .. $23

**COMMODORE**
Easy File (D) ... $75
Easy Finance (D) ... $38
Easy Mail (D) ... $38
Easy Script (D) ... $75
Easy Schedule (D) ... $59
Logo (R) ... $75
Pilot (D) ... $75
Assembler (D) ... $38
Music Machine (D) ... $25
Music Composer (D) .. $25
Meza Music (D) ... $75
Video/Music Supt. (D) $38
Jupiter Lander (R) ... $25
Radar Rat Race (R) ... $25
Sea Wolf (R) ... $25
Kickman (R) ... $25

**COMM-DATA**
Pakacuda . (C) $14 (D) $18
Escp. MCP . (C) $14 (D) $18
Centropods (C) $14 (D) $18

**COMPUTERMAT**
Arcade-Pak (C) ... $18
Education-Pak (C) ... $18

**CREATIVE SOFTWARE**
Moondust (R) ... $25
Trashman (R) ... $25
Save New York (R) ... $25
Astroblitz (R) ... $25
Household Fin. (D) ... $25

**DATA 20**
Video Pak 80 ... $139
Z80 Video Pak ... $229

**EN-TECH**
Finance Calc 64 ... $34
Data Base 64 ... $56
Invoice Ease 64 ... $56

**EPYX**
Temple of APS (D) ... $27
Upper Reach. APS (D) $14
Jumpman (D) ... $27

**HES**
HES Modem ... $59
6502 Prof.Dev.Sys.(C) . $22
Hesmon 64 (R) ... $27
Turtle Grapics II (R) .. $41
Heswriter 64 (R) ... $32
Gridrunner (R) ... $27
Retroball (R) ... $27

**INFOCOM**
Zork I, II or III (D) ... $27
Deadline (D) ... $35
Starcross (D) ... $27

**JIN SAM**
Mini-Jini (R) ... $75

**LITTLE WIZARD**
Pro.Mail.List (C)$22 (D)$25
Stockmaster (Inventory) (C)$25 (D)$28

**LOGISTIC**
Datacalc 64 (C) $55 (D) $59
Home Journal (D) ... $55

**MICROSPEC**
Payroll System (D) ... $73
Inventory Pkg (D) ... $73
General Ledger (D) ... $73
Disk Data Mgr (D) ... $62
Mail List Mgr (D) ... $41
Checkbook Mgr (D) ... $39

**M-SOFT**
M-File (D) ... $89

**ON-LINE**
Frogger (D) ... $23
Jawbreaker (D) ... $20

**PACIFIC COAST SOFT.**
PCS (80 Col BD, Word Proc, D.Base,Spreadsheet) CALL
Account PAC (C/D) ... $34
File PAC (D) ... $30
Editor PAC (D) ... $39
Inquire PAC (D) ... $57
Happy Tutor Typng (D) $18

**PROFESS. SOFTWARE**
Wordpro 3 + /64 (D) ... $68

**QUICK BROWN FOX**
Prof.Word Proc. (R) ... $50

**RAINBOW**
Writers Assistant ... $95
Spreadsheet Assist. .. $95
File Assistant ... $95

**SIRIUS**
Blade/Blackpoodle (D) $27
Type Attack (D) ... $27
Repton (D) ... $27
Critical Mass (D) ... $27
Snake Byte (D) ... $23
Way Out (D) ... $27
Fast Eddie (D) ... $23
Turmoil (D) ... $23
Spider City (D) ... $27
Squish'Em (D) ... $23
Final Orbit (D) ... $27
Alpha Shield (D) ... $27

**SKYLES ELEC. WORKS**
Busicalc (C/D) ... $52
Busiwriter (D) ... $72

**SPINNAKER**
Snooper Troops 1 (D) . $29
Facemaker (D) ... $23
Kindercomp (D) ... $20
Hey Diddle (D) ... $20
Most Amaz. Thing (D) . $27

**SYNAPSE**
Fort Apocalypse (C/D) $23
Survivor (C/D) ... $23
Drelbs (C/D) ... $23
Pharoh's Curse (C/D) . $23
Protector II (D) ... $23
Morgal (D) ... $23
Shamus (D) ... $23

**TAYLORMADE**
Touch Typing Tutor 3.0 (D) ... $21

**TIMEWORKS**
Rbbrs/Lost Tomb (C/D) $21
Wall Street (C/D) ... $21
Money Manager (C/D) $21
Data Master (C/D) ... $21
Dungeons of Alg. Dragons (C/D) ... $21

**TOTL**
Text 2.6 ... (C) $32 (D) $34
Label 2.6 .. (C) $15 (D) $17
Time Manager 2.6 (C) . $24
Time Manager 2.6 (D) . $27
Resrch Assist. 2.0 (C) . $24
Resrch Assist. 2.0 (D) . $27

**UMI**
Motor Mania (C) ... $20
Renaissance (C) ... $27

**VICTORY**
Annihilator (C/D) ... $16
Kongo Kong (C/D) ... $16
Trek (C/D) ... $14
Adv. Pack #1 (C/D) ... $16
Adv. Pack #2 (C/D) ... $16
Grave Robbers (C/D) .. $13
Chomper Man (C/D) .. $18

# 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
```

©

# Extra Instructions

Joel C. Shepherd

*Combining machine language instructions by creating new opcodes can result in more efficient programming in certain situations. Machine language programmers can conserve memory and increase execution time with the methods discussed here. This article explores the unofficial, hidden "commands" you can give the 6502.*

The 6502 microprocessor can execute 151 instructions. There are 56 different types of instructions in its library, and each can be used with at least one of the 13 addressing modes available. Each instruction performs one operation, such as loading a register, setting a flag, or rotating a byte, and deals with one byte of data or one memory location. The 6502's machine language consists of nothing more than variations on these simple instructions. It is an extremely low-level language which is both a boon and a bane to programmers. While machine language is versatile and applicable to any programming problem, it can also be tedious in a particular sense. It can require programmers to use several instructions of the same type performing similar, yet separate, operations.

Consider the LDA and LDX instructions; both load a register and set the same processor status flags. They also share four addressing modes. One, however, loads the accumulator and the other loads the X register. There is no instruction which will load both. The language is so simple that it has no instructions with more than one memory location or register as an operand.

There is no reason why the 6502's instruction set cannot be expanded to include more sophisticated instructions. Such expansion, however, would probably also drive up the processor's cost. The designers of the 6502, in an effort to keep its cost low, decided to give it a simple machine language.

## Creating New Opcodes

A possible solution to the problem would be to "trick" the processor into executing instructions it didn't "know." Imagine being able to load two registers simultaneously or being able to AND two registers with one instruction. There is a simple way to do just that and more.

If you examine a list of 6502 opcodes, you'll notice that none of the codes have the form a7 or aF (a is any hexadecimal digit). Codes of this form are not "official" 6502 opcodes, yet all are executable and potentially useful.

An example is opcode AF. Opcodes AD and AE are the LDA $aaaa and LDX $aaaa instructions respectively. Opcode AF combines the two: it loads the accumulator and the X register from the same location. For instance, these two instructions:

```
AD 00  16  LDA   $1600    ;load accumulator
AA         TAX            ;and X
```

can be replaced by:

```
AF 00  16  LDAX  $1600    ;load A and X
```

(LDAX is a new mnemonic. I've included mnemonics to represent the operations performed by the new instructions.) Using LDAX saves one byte and two cycles of execution time. That might not seem like much, but in a loop routine the time savings may be significant. (The execution times given for the routines using the new opcodes are estimates. More on that later.) Additionally, with this instruction, it is possible to load the accumulator from a zero-page location indexed by the Y register; this is not possible with the LDA instruction.

Opcode 87 is another example. Opcode 85 is the STA $aa instruction, and 86 is the STX $aa instruction. Opcode 87 executes both 85 and 86, but with a twist. It is impossible for a location to contain two different values at one time. Opcode 87 logically ANDs the accumulator with the X register, changing neither, and stores the result in the address specified. In other words, only the bits which are set in both the accumulator and the X register are set in the byte which is stored. So, the routine:

```
A5  AD     LDA   $AD      ;get char
87  AD     ANDX  $AD      ;mask with X
```

## Summary Of Extra 6502 Instructions

| Name/Description | Operation | Addressing Mode | Machine Language Form | Hex Op-Code | N | # | NZCV |
|---|---|---|---|---|---|---|---|
| ANDX<br>AND A with X | A&X → M | Zero Page | ANDX $aa | 87 | 3 | 2 | ---- |
| | | Zero Page,Y | ANDX $aa,Y | 97 | 4 | 2 | |
| | | Absolute | ANDX $aaaa | 8F | 4 | 3 | |
| DCMP<br>Decrement M,<br>Compare to A | A-(DEC M) | Zero Page | DCMP $aa | C7 | 8 | 2 | ???- |
| | | Zero Page,X | DCMP $aa,X | D7 | 10 | 2 | |
| | | Absolute | DCMP $aaaa | CF | 10 | 3 | |
| | | Absolute,X | DCMP $aaaa,X | DF | 11* | 3 | |
| ISBC<br>Increment M,<br>Subtract from A | A-(INC M)-C → A,C | Zero Page | ISBC $aa | E7 | 8 | 2 | ???? |
| | | Zero Page,X | ISBC $aa,X | F7 | 10 | 2 | |
| | | Absolute | ISBC $aaaa | EF | 10 | 3 | |
| | | Absolute,X | ISBC $aaaa,X | FF | 11* | 3 | |
| LDAX<br>Load A and X | M → A,M → X | Zero Page | LDAX $aa | A7 | 3 | 2 | ??-- |
| | | Zero Page,Y | LDAX $aa,Y | B7 | 4 | 2 | |
| | | Absolute | LDAX $aaaa | AF | 4 | 3 | |
| | | Absolute,Y | LDAX $aaaa,Y | BF | 4* | 3 | |
| RLAN<br>Rotate M left,<br>AND with A | (ROL M) &A → A | Zero Page | RLAN $aa | 27 | 8 | 2 | ???- |
| | | Zero Page,X | RLAN $aa,X | 37 | 10 | 2 | |
| | | Absolute | RLAN $aaaa | 2F | 10 | 3 | |
| | | Absolute,X | RLAN $aaaa,X | 3F | 11* | 3 | |
| RRAD<br>Rotate M right,<br>Add with carry | (ROR M) + A + C → A,C | Zero Page | RRAD $aa | 67 | 8 | 2 | ???? |
| | | Zero Page,X | RRAD $aa,X | 77 | 10 | 2 | |
| | | Absolute | RRAD $aaaa | 6F | 10 | 3 | |
| | | Absolute,X | RRAD $aaaa,X | 7F | 11* | 3 | |
| SLOR<br>Shift M left,<br>OR with A | (ASL M)V A → A | Zero Page | SLOR $aa | 07 | 8 | 2 | ???- |
| | | Zero Page,X | SLOR $aa,X | 17 | 10 | 2 | |
| | | Absolute | SLOR $aaaa | 0F | 10 | 3 | |
| | | Absolute,X | SLOR $aaaa,X | 1F | 11* | 3 | |
| SREO<br>Shift M right,<br>Exclusive OR with A | (LSR M)V̶ A → A | Zero Page | SREO $aa | 47 | 8 | 2 | ???- |
| | | Zero Page,X | SREO $aa,X | 57 | 10 | 2 | |
| | | Absolute | SREO $aaaa | 4F | 10 | 3 | |
| | | Absolute,X | SREO $aaaa,X | 5F | 11* | 3 | |
| TSTA<br>Test bit 2 in A | A&#$04 → A | Absolute | TSTA $aaaa | 9F | 4 | 3 | ---- |
| TSTX<br>Test bit 2 in X | X&#$04 → A | Absolute | TSTX $aaaa | 9E | 4 | 3 | ---- |

The following notation applies to this summary:

| | | | | | |
|---|---|---|---|---|---|
| A | Accumulator | - | No Change, or Subtract | V | Logical OR |
| X,Y | Index Registers | + | Add | $aa | 8-bit Zero-Page Address |
| M | Memory | & | Logical AND | $aaaa | 16-bit Absolute Address |
| C | Carry Flag | V̶ | Logical Exclusive OR | N | Number Of Clock Cycles |
| ? | Change | → | Transfer To | # | Number Of Storage Bytes |
| | | | | * | Add 1 if page boundary is crossed |

# Complete Personal Accountant: we've made the best much more friendly.

If you have any doubts that we offer the best and most complete personal financial package available, look over the features listed below. Now we have the only package with **full screen editing** for Atari 400/800,* TRS-80 COLOR, Commodore 64* and VIC-20; the ability to move the cursor in any direction makes our accountant-designed package even more friendly than before. No one else offers all of these:

1. CHECKBOOK MAINTENANCE—automatically balances your checkbook with each entry; manages checks, charges, deposits, and interest quickly and accurately.
2. CHART OF ACCOUNTS—maximum of user flexibilty with up to 99 accounts plus 9 subcategories may be defined.
3. CHECK SEARCH—multi-reference; tracks items on every field including tax deductibles.
4. NET WORTH/ INCOME/EXPENSE STATEMENT—know-exactly-where-you-stand program generates statements with the touch of a key.
5. DETAIL & SUMMARY BUDGET ANALYSIS—an absolute necessity in financial planning.
6. CHECK WRITER—prints personalized checks.**
7. PAYMENTS/APPOINTMENTS CALENDAR—monthly displays of up to 250 bills and 200 appointments.
8. COLOR GRAPH DESIGN PACKAGE—graphs all monthly files.
9. MAILING LIST—maintains all records, sorts by name or zip, allows add/change/delete.
10. FRIENDLY USER MANUAL—complete with indexing, flow charts and diagrams; the most thorough documentation on the market.

This all adds up to the finest personal financial system available—comprehensive enough for a small business. Less than one hour of data input per month will allow this **menu-driven package** to help you handle your finances with a lot more fun than drudgery.

Plus, ours is the only expandable system; purchase the package in sections and add on as your financial needs grow. Features 1, 2, 3 and 6: $39.95 diskette, $36.95 cassette; Features 4 and 5: $29.95 diskette, $26.95 cassette; Features 7, 8 and 9: $29.95 diskette, $26.95 cassette; **or save $19.90 or $15.90 respectively by buying the entire system for $79.95 diskette, $74.95 cassette.**


*Random Access [illegible]

Prices subject to change without notice. See your local dealer or order direct. New catalog available.
Add $3.00 for postage and handling. Credit card orders call toll free:

# 1-800-334-SOFT

**DEALER INQUIRIES INVITED**


# programmer's institute

a division of **future**house

p.o. box 3470, dept. C, chapel hill, north carolina 27514, 919-967-0861

(ANDX is a mnemonic for 87) loads the accumulator, ANDs it with the X register, and stores the result in location $AD. It occupies four bytes and takes six cycles to execute. Compare that to the following equivalent routine:

```
8A          TXA          ;X into A
25   AD     AND   $AD    ;AND "X" with $AD
85   AD     STA   $AD    ;store result
```

which requires five bytes and eight cycles to execute. One byte and two cycles are saved by using the first routine.

Opcode DF is an interesting instruction. Opcode DD compares the accumulator to an X indexed address, and opcode DE decrements the value in an X indexed address by one. Opcode DF first decrements the value in the address given and then compares the result to the accumulator. For example:

```
A9   A5        LDA   #$A5    ;loop terminator
DF   00   16   DCMP  $1600   ;done?
D0   aa        BNE   $....   ;no, do next
```

(DCMP is a mnemonic for DF) loads the accumulator with #$A5, decrements the value in $1600 by one, compares $1600 to the accumulator, and branches if they are not equal. This could be used to control a loop.

## How The New Opcodes Execute

In the table, I have included ten new instruction types, creating 41 new instructions. The operations performed by the new instructions are combinations of the "original" 6502 instructions. There are some general rules for predicting what one of the new opcodes will do.

First, all of the new instructions execute the two preceding instructions. Opcode DF executes both DD and DE, for example. The two preceding instructions we'll call the "sub-instructions" of the new instructions. So, a5 and a6 are sub-instructions of a7, and aD and aE are aF's sub-instructions.

If the sub-instructions are different types of instructions, the sub-instruction with the greatest opcode will be executed first; otherwise the sub-instructions will be executed simultaneously. For instance, since LDA and LDX are both load instructions, LDAX will execute them simultaneously. But, since ASL and ORA are different types of instructions, ASL (opcode 06) will be executed first, followed by ORA (05) when opcode 07 is executed. There is a definite inconsistency in this rule: it implies that the processor examines the current instruction and then decides to execute the sub-instructions either one at a time or simultaneously. The 6502 isn't that sophisticated. The rule, however, is the only one that explains the operations of the new opcodes. The given execution times are based on this rule.

Generally, the same operand is used for both sub-instructions. Consider:

```
07   19        SLOR  $19
```

(SLOR is a mnemonic for 07). This ASLs $19 and then ORAs $19 with the accumulator. Location $19 is the operand for both sub-instructions.

Opcodes 97, B7, and BF are special cases of the preceding rule. All three have sub-instructions which use different addressing modes, X indexed and Y indexed. The operands used by the new instructions are the given addresses indexed by the Y register. For instance, B5 is the LDA $aa,X instruction and B6 is the LDX $aa,Y instruction. Opcode B7, then, is the LDAX $aa,Y instruction.

## Two Exceptions

There are two additional opcodes which are exceptions to all of the rules discussed: 9E and 9F. Opcode 9E logically ANDs the X register with the value #$04, keeping the X register intact. The result is stored in an absolute memory location. For example:

```
A6   F6        LDX   $F6
9E   40   03   TSTX  $0340   ;is bit 2 set?
```

(TSTX is a mnemonic for 9E) loads the X register, ANDs it with #$04 and stores the result in $0340. The origin of the #$04 is unknown.

Opcode 9F is similar to 9E: the accumulator is ANDed with #$04 and the result is stored in an absolute location. So, these instructions:

```
A9   06        LDA   #$06
9F   00   00   TSTA  $0000
```

(TSTA is a mnemonic for 9F) AND the accumulator with #$04 (yielding #$04) and store the result in $0000. The accumulator will contain #$06 when this routine is finished.

The execution times of the new instructions in the table are based on whether the sub-instructions are of the same type or not.

It seems that one of the major shortcomings of the new instructions is their specialization. For example:

```
4A   aa        LSR   $aa
45   aa        EOR   $aa
```

is probably not used in too many programs. The SREO (mnemonic) instruction (which replaces the above instructions) is too specialized to be of much use to anybody.

Currently, there are no assemblers which accept the new mnemonics as valid instructions. The opcodes, however, can be POKEd directly into the computer. An interesting, and useful, project would be to write an assembler which could handle the new instructions. It wouldn't be difficult to modify an assembler written in BASIC to recognize these new mnemonics. ©

# The Light Pen at the Right Price:


Shown actual size.

**$34.95** Each

This is the **right** pen—a truly affordable peripheral. As an education aid to young children it is without equal, especially for pre-schoolers without keyboard skills. They simply touch the screen with the pen and a display is altered or new information is entered. Plus as an aid to games, it offers comparable utility to a joystick. Create your own programs with The Light Pen or choose from our software which includes a simple draw routine, checkers and hangman; or backgammon and chess. The Right Pen at the right price—maximum utility and minimum cost.

Prices subject to change without notice. See your local dealer or order direct. New catalog available.
Add $3.00 for postage and handling. Credit card orders call toll free:

**1-800-334-SOFT**

**DEALER INQUIRIES INVITED**

**programmer's institute**

a division of **futurehouse**

p.o. box 3470, dept. C, chapel hill, north carolina 27514, 919-967-0861

# Commodore DOS Wedges: An Overview

Jim Butterfield, Associate Editor

*Most Commodore machines have a small program called a "DOS Wedge" which allows convenient use of various disk commands. Recently, copies of this program have been in circulation without information; many users who have it don't know how to use it. Here are the details.*

## What It Does

DOS Wedge programs provide for three types of capability:

- the disk's error status may be checked with a simple command;
- a disk directory (or catalog) may be obtained without disturbing a program which has already been loaded;
- a number of "disk commands" may be easily transmitted to the disk.

## How To Start The Wedge

Wedge programs are often self-relocating. If you LOAD the wedge program and say RUN, the program will find a spare place in memory and pack the DOS wedge system there. After this, you may type NEW and the wedge will remain in place: it's been packed into a safe part of memory.

Some wedge programs come in "pieces," or as a set of programs on the disk. The first program, called the "boot," loads in the others as necessary. This is usually the case with the C-64 Wedge, for example.

Commodore 64 and VIC-20 wedge programs often also come with a how-to program, explaining the system and how it works. Many PET/CBM wedge programs print brief instructions on the screen as they self-relocate.

For other systems, you may find it necessary to use the last resort of reading the instructions.

## First Rules In Using The Wedge

DOS Wedge commands must be typed in as direct commands. You cannot include them as part of a program. This is deliberate: programs will run at full speed without the need to check for the extra wedge commands.

Wedge commands should start in column 1 of the screen. The primary wedge commands are flagged with either of two symbols:

> or @

The "greater than" sign was the original. It looks like a wedge and is partly responsible for giving the wedge program its name. On later Commodore computers, this character needed the shift key; to save finger work, the @ (at sign) character was allowed as an alternative.

A few supplementary commands which use other starting symbols are available. They tend to be less important, since their functions can be performed easily with conventional commands. The most common of these are:

/ – to load a program; and
↑ – to load and run a program.

Thus, a command such as /DOGLEG will cause a program named DOGLEG to be loaded from disk.

## Pattern Matching

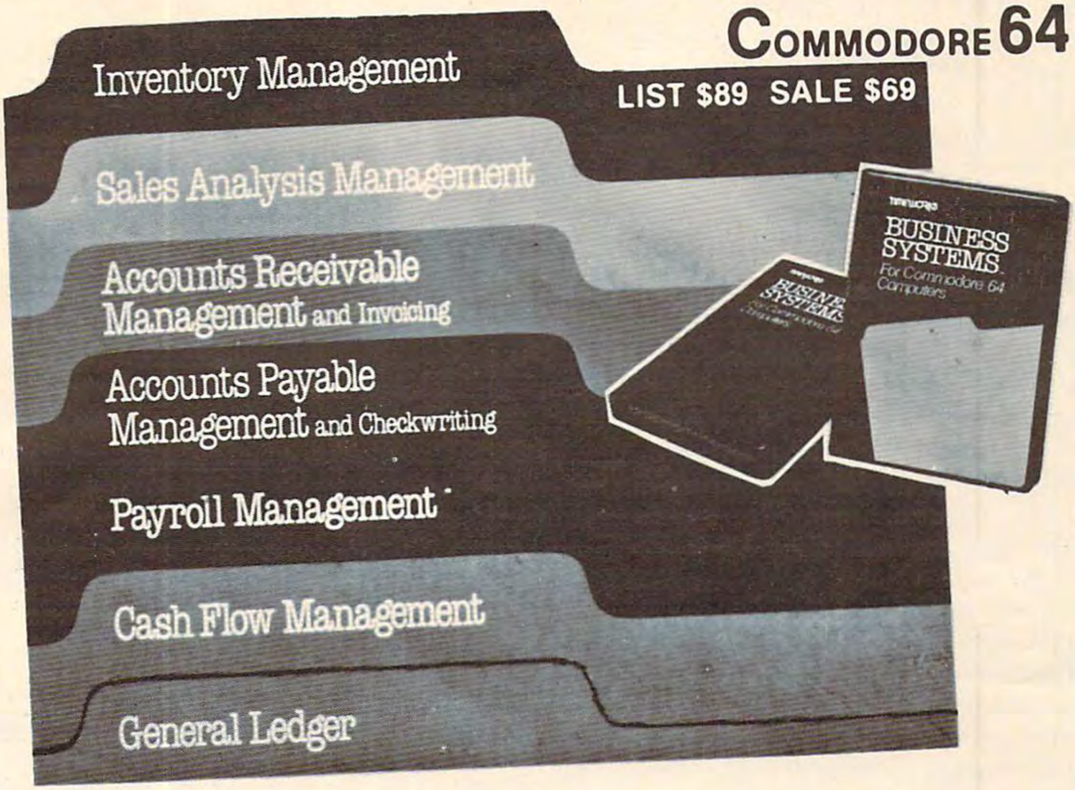
In many cases, a program name does not have to be specified exactly. We can give part of the name and use "pattern matching" to find one or more programs that match the name. The pattern matching characters are:

? – to match any single character; and
* – to match any stream of characters.

Thus, a command such as /DOG* will load the

# PROFESSIONAL BUSINESS SOFTWARE SALE!

## COMMODORE 64




## Executive Quality by TIMEWORKS™
## "Cadillac of Business Programs"

## Each Program comes in a beautiful case and includes: *Disk*

- A Menu-driven program, sophisticated enough to provide complete Management and Product information, yet requires no prior computer or accounting knowledge to operate.
- A Manual written in easy-to-understand, people-friendly English, abundantly illustrated to provide further clarity and eliminate guesswork.
- A unique method of creating your own, unlimited array of reports—easily and quickly. YOU select the information you want, and YOU determine the sequence of the report column headings.
- Password Protection, to prevent unauthorized access to confidential data.
- A program wnich can be used by itself (stand-alone), or can be interfaced, one at a time, with other TIMEWORKS MANAGEMENT INFORMATION programs into a fully integrated accounting system.

### PLUS: THESE PROFESSIONAL BUSINESS PROGRAMS ·

- ***Script 64 · No. 1 Executive Word Processor · Disk*** — ***List $99 · Sale $59***
- ***Complete Data Base · Tape-Disk*** — ***List $89 · Sale $59***
- ***Electronic Spread Sheet (like Visicalc) · Tape-Disk*** — ***List $89 · Sale $59***

**• LOWEST PRICES • 15 DAY FREE TRIAL • 90 DAY FREE REPLACEMENT WARRANTY**
**• BEST SERVICE IN U.S.A. • ONE DAY EXPRESS MAIL • OVER 500 PROGRAMS • FREE CATALOGS**

Add $3.00 for postage. Add $6.00 for CANADA, PUERTO RICO, HAWAII orders. WE DO NOT EXPORT TO OTHER COUNTRIES.

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

## PROTECTO
**ENTERPRIZES** (WE LOVE OUR CUSTOMERS)

**BOX 550, BARRINGTON, ILLINOIS 60010**
**Phone 312/382-5244 to order**

first program it comes to whose name starts with the characters DOG. This might include DOGLEG, DOGHOUSE, DOG, and DOGOOD. Similarly, a command such as /D?G will load a three-character name such as DIG, DOG, or DRG.

Pattern matching can be used for many DOS Wedge commands. You must not use it, of course, when you SAVE a program, since the program must have an exact name.

## Disk Status

This is perhaps the most useful (and the briefest) DOS wedge command of all. Simply type the › or @ character, followed by the RETURN key, and you'll get the status of the disk.

Most of the time, the disk will not report an error, but will give you a message like: 00,OK,00,00. If there's an error, or immediately following a SCRATCH command (more on this later), you'll get a different message. Following a SCRATCH, you'll be told how many files have been removed from disk. If the disk is signalling an error condition, you'll be told what the error is.

Most of the time, the error will be an obvious one, and you'll spot the difficulty right away. You might have attempted to write to a protected disk, or one that is full. You might have tried to create a new file giving a name that already exists. You might be trying to read or load a file that isn't there. You won't need to translate the number if the disk reports 62,FILE NOT FOUND,00,00. Sometimes, however, the disk error number can give useful information; look it up in the manual.

4.0 Equivalent: Users with a 4.0 BASIC system can type PRINT DS$ and receive the equivalent information.

Non-wedge Equivalent: If you don't have a wedge, you'd need to write a program to open secondary address 15 and INPUT from that channel. (INPUT works only from within a program.) Get a wedge.

## Directory Or Catalog

You may obtain a catalog from disk, without disturbing a program already loaded, by typing ›$ or @$. If you have a dual disk (two drives ganged together in a single housing), you may specify the drive by adding a number, yielding commands such as ›$0 or @$1. Many experienced users suggest that you'll do well to specify drive 0 for single disks: the zero in @$0 won't hurt and might help.

You may take a "specific" catalog by using a filename with pattern matching. For example, ›$0:D* will give only programs whose names start with the letter D. @$0:??? will report programs with exactly three characters in their names.

4.0 Equivalent: The command CATALOG or DIRECTORY will produce a directory. There's no simple way to get a pattern match with this.

Non-wedge Equivalent: You'll need to get a directory using a LOAD "$0",8 command. This destroys the program you have in place. (Get a wedge.)

## Other Disk Commands

By following the › or @ sign with command characters, you may send a number of special commands to the disk. We'll list them below.

4.0 Equivalent: Specific commands are often available, such as SCRATCH, COLLECT, or HEADER. We'll deal with them individually below.

Non-wedge Equivalent: You must do this by opening secondary address 15, and then sending the command as part of a PRINT# sequence. For example, ›I0 could be matched with OPEN 15,8,15:PRINT#15,"I0". This is no hardship since it can be done with direct commands. The wedge is slightly more convenient.

## Initialize

The command ›I0 or @I1 causes the disk electronics to "shake hands" with the appropriate disk drive for a smoother transfer of data. It shouldn't be needed, but can be most useful if you start encountering DRIVE NOT READY errors. On single disks, @I0 is recommended.

4.0 Equivalent: None. You'd have to use the secondary address 15 equivalent.

## Header (Or New)

Watch this one: It's powerful, and will wipe the previous contents of a disk. @N0:DISKNAME,JB will format the drive over all its tracks, using the ID of "JB", and will prepare an empty directory with the title "DISKNAME". This job will take a few minutes.

@N0:ANOTHER will not format the disk. Instead, it will just wipe the directory and put the new title in place, in this case "ANOTHER". It only takes a couple of seconds to do this, but all your disk information will immediately disappear. The disk must have previously been formatted before this version of the command can be used.

You cannot use a disk until it has been formatted. Be very careful that all your working disks have different IDs (the two letter identifiers); this will give your disk-stored programs and files important protection against harm.

4.0 Equivalent: HEADER. The 4.0 system asks ARE YOU SURE? before it dives in and wipes the data from the disk.

## Scratch

Removes one or more files from a disk and frees the directory and disk space. A command of @S0: TURNPIKE removes the file called TURNPIKE, if there is one by that name on the disk.

# DISK CLEANER

## Everyone Needs A

## Disk Drive Doctor

# Sale $19.95

### FACTS

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

**The Cheetah disk drive cleaner can be used with single or double sided 5½" disk drives. The Cheetah is an easy to use fast method of maintaining efficient floppy diskette drive operation.**
**The Cheetah cleaner comes with 2 disks and is packed in a protective plastic folder to prevent contamination. List $29.95 / Sale $19.95**

| Features | Cheetah | 3M | Discwasher | Verbatim |
|---|---|---|---|---|
| Cleanings Per Kit | **40 +** | 25 | 30 | 2 original - 10 refill |
| Price | **$19.95** | $34.95 | $24.95 | $12.95 |
| Abrasivity | **None** | Minimum | High | Moderate |
| Programming Required | **No** | No | Yes | No |

Add $3.00 for postage. Add $6.00 for CANADA, PUERTO RICO, HAWAII orders. WE DO NOT EXPORT TO OTHER COUNTRIES.

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

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

You may use pattern matching, but watch it! A command of ›S0:C* will remove all files which start with the letter C. A command of @S0:* will remove all files, giving the same effect as New, but slower.

4.0 Equivalent: SCRATCH.

## Verify (Validate, Collect)

A widely misunderstood command. This causes the disk unit to try to rebuild its map of free blocks, removing incomplete files as it does so. It does not report errors; it just reconstructs the Block Availability Map (BAM), using information from the directory.

If you have somehow created an incomplete file, it will appear in the directory with an asterisk beside the file type (e.g., *SEQ). You should immediately remove it with @V0 in order to avoid complications. (*Do not, repeat do not, try to scratch such a file.*) You'll probably need to wait awhile: rebuilding the BAM often takes time.

4.0 Equivalent: COLLECT.

## Rename (Name)

Lets you change the name of a file. To change WATER to ICE, you type @R0:ICE=0:WATER. Think of the syntax as similar to BASIC: the value to the left of the equals sign indicates what has been changed.

4.0 Equivalent: RENAME ... TO ...

## Copy

Lets you make a new copy of a program on disk. Most useful on dual disk units, but you can make an extra copy even when you have a single disk. A command such as C0:DOG=0:CAT will make an extra copy of file CAT, naming the new copy DOG.

A lesser-known feature of the Copy command is that it allows two or more files to be put together (or "concatenated") into a single file. This can be done with a command such as @C0:FRACAS=0:DOG,0:CAT which puts files DOG and CAT together into a single new file called FRACAS. This feature is most useful for data files, by the way; you *can* stick programs together, but the result won't be very useful.

4.0 Equivalent: COPY ... TO ...; or CONCAT ... TO ...

## Duplicate (Backup)

Only for dual drives. Allows a disk to be copied, block for block, from one drive to the other. Dangerous! Be sure you specify the correct direction, or you're in big trouble. So @D1=0 will make an exact copy of the information on drive 0 placing it onto drive 1. If you intended to go the other way, it's too late.

4.0 Equivalent: BACKUP D0 to D1. The system does not ask ARE YOU SURE? It just goes ahead.

## Other Commands

There are other commands that may be sent down the command channel, but I don't recommend their use with a wedge. In fact, I recommend careful study of the manual or sample programs before using them in any form.

There are several DOS wedges; it's impossible to cover details on each of them. But they have similar patterns.

Learning to use the DOS wedge effectively is almost the same as learning to use the disk effectively. More correctly, it's learning how to use the disk – easily and effectively. ©

# SJB DISTRIBUTORS. THE MOST COMPETITIVE PRICES ON COMMODORE.

## commodore

### NEW COMMODORE PRODUCTS

| | |
|---|---|
| The Executive 64 | Call |
| CBM C128-80 | $ 795 |
| CBM BX700 | 2990 |
| B Series Software | Call |
| CBM 1520 Plotter | 169 |
| CBM 1526 Printer | 349 |

### SOFTWARE FOR CBM 64

#### BUSINESS

| | |
|---|---|
| Word Processing (WordPro 3+) | $ 69 |
| Quick Brown Fox | 56 |
| Writers Assistant | 99 |
| Spell Master | 75 |
| Calc Result | 125 |
| Busicalc II | 95 |
| Spread Sheet Assistant | 99 |
| Data Manager | 70 |
| M File (merge with WordPro) | 89 |
| Info Mast | 139 |
| 64 Mailing List | 28 |
| The Manager | 50 |
| Home Accountant (continental) | 75 |
| Finance Assistant | 45 |
| Stock (investment analysis) | 80 |
| Agricultural Management | Call |
| General Ledger, A/R, A/P, P/R, Inv | Call |

#### RECREATION

| | |
|---|---|
| Assembler Package (cassette or disk, compiled, includes editor, loader, disassembler) | 39 |
| Sprite Master (access) | 30 |
| Neutral Zone (access) | 35 |
| Space Belt | 19 |
| Pet Emulator | 30 |
| Coco II (build your own games) | 40 |
| Vic Tree (programmers utilities) | 75 |
| Micro-Term (save to printer,disk) | 39 |
| Hesmon | 35 |
| Synthesound | 45 |
| Gothmogs Lair | 30 |
| Road Toad | 15 |
| Commodore Games | Call |

### INTERFACES & ACCESSORIES

| | |
|---|---|
| 80 Column Expander | $159 |
| VIC 1600 Modem | 95 |
| VIC 1650 (auto answer, auto dial) | 150 |
| VIC 1525 Graphic Printer | 225 |
| VIC 1530 Datasette Recorder | 65 |
| VIC 1541 Disk Drive | 249 |
| VIC Switch (connect 8 64's or Vics to printer, dd) | 149 |
| PET-IEEE cable | 33 |
| IEEE-IEEE cable (2m) | 49 |
| Parallel Interface (Epson, Okidata, IDS, NEC) | 70 |
| Programmers Reference Guide | 18 |
| Verbatim Diskettes (10 per box) | 26 |
| Hes Modem | 75 |
| ADA 1450 | 149 |
| ADA 1800 (new) | 129 |
| Numeric Keypad | 65 |

### VIC PRODUCTS & ACCESSORIES

| | |
|---|---|
| 8K RAM Memory Expansion Cartridge | $ 40 |
| 16K RAM | 70 |
| 24K RAM | 105 |
| VIC 3 Slot Expander | 27 |
| VIC 6 Slot Expander | 70 |
| Cassette Interface | 30 |
| Gorf (64 also) | 30 |
| Omega Race | 30 |
| Arcade Joystick - Heavy duty w/2 firing buttons! Great for the VIC or 64 | 25 |
| Auto Clock | 125 |

### MONITORS - GREAT RESOLUTION (64 OR VIC)

| | |
|---|---|
| CBM 1701 Color Monitor | $ 249 |
| Amdek Color Plus | 299 |
| Panasonic TR-120 (w/speaker) | 155 |
| Panasonic CT-160 | 279 |
| BMC (green screen) | 95 |
| Transtar 20 (high resolution green phosphor) | 129 |
| Video/Audio Cable | 15 |

### PRINTERS - LETTER QUALITY

| | |
|---|---|
| CBM 6400, 40 cps | $1450 |
| Diablo 620, 25 cps | 949 |
| Transtar 140 (serial) | 1395 |
| Transtar 130, 16 cps (auto load, wp features!) | 769 |
| NEC 3500 Series | 1600 |
| NEC 7700 Series | 2350 |

### PRINTERS - DOT MATRIX

| | |
|---|---|
| CBM 8023, 150 cps/graphics | $ 545 |
| CBM 4023 Printer | 395 |
| Epson FX Printer, 160 cps | 549 |
| Epson MX-80 FT w/graftrax | Call |
| Epson FX-100 | 859 |
| Okidata 82A, 120 cps (serial and parallel) | 429 |
| NEC 8023A (parallel) | 429 |
| Okidata 92 | 559 |
| Star Gemini, 10 | 329 |
| Star Gemini, 15 | 499 |
| Transtar 315 (hi-res, color) | 575 |

### COMMODORE BUSINESS SERIES

| | |
|---|---|
| SuperPet (5 languages, 2 processors) | $1059 |
| CBM 8032 Computer, 80 Column | 595 |
| CBM Memory Expansion, 64K | 259 |
| CBM 8050, 1 mg. Dual Drive | 995 |
| CBM 8250, 2 mg. Dual Drive | 1295 |
| CBM D9060, 5 mg. Hard Disk | 1995 |
| CBM D9090, 7.5 mg. Hard Disk | 2250 |
| CBM 2031, 170K Single Drive (New) | 395 |
| DC Hayes Smart Modem | 220 |

### BUSINESS SOFTWARE-8032

| | |
|---|---|
| WordPro 4+ or 5+ | $ 309 |
| InfoPro | 219 |
| Administrator | 489 |
| VisiCalc (expanded) | 199 |
| BPI A/R, G/L, Job Cost, Inventory, Payroll | ea.325 |

**MasterCard, Visa, Money Order, Bank Check**

COD (add $5) accepted.
Add 3% surcharge for credit cards.
In stock items shipped within 48 hours.
F.O.B, Dallas, Texas
All products shipped with manufacturer's warranty.
Prices are subject to change without notice.

**TO ORDER**
**CALL TOLL FREE**
**800-527-4893**
**800-442-1048**
(Within Texas)
Business Hours
Mon.- Fri. 8 to 6, Sat. 10-2

Write for free catalog.

### GAME OF THE MONTH

Adventu-Writer (make your own adventure games) Join the Adventu-Writer Club ............. $ 49

### PRODUCT OF THE MONTH

INTERPOD (intelligent IEEE RS232, serial interface for VIC or C64) ............. $ 179

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

# Protector For VIC-20

George Trepal and Doug Smoak

*Disabling certain commands on the VIC can be a very useful way to protect your programs. This article demonstrates how to disable the SAVE, RESTORE, STOP, and LIST commands with a few simple POKEs. There are some techniques here for the 64, too.*

Sometimes it's nice to have a VIC that isn't fully functional. Maybe you're a teacher and you don't want your program listed by students. Or perhaps you've written a game you don't want someone to save. Let's look at a few ways to block your VIC's SAVE, RESTORE, STOP, and LIST.

Obviously, it's nice to have a program that can't be SAVEd (that is, copied). When you SAVE a program, the VIC goes to a *table* (a list of memory addresses in RAM) to get the address of the SAVE routine. Once it has the address, it jumps to the routine and puts the program on tape. Guess what happens if it gets the wrong address.

If you POKE the two values under the protection method for SAVE, the address now points to the LOAD routine. When you tell the VIC to SAVE, it tries to LOAD instead. It would have been just as easy to give the address of the NEW or LIST routine or anything else. In fact, as long as the value of 818 is not 113, or the value of 819 is not 246, the machine will not SAVE. You can try any numbers you want and see what happens. To reactivate the SAVE routine, POKE in the normal values in the table. Or you can press the RUN/STOP and RESTORE keys at the same time, and that will reassign the correct addresses.

Now let's get the RESTORE key out of action so that it can't correct the wrong address. Check the table and use the two POKEs. Notice that the table doesn't give the normal values for the RESTORE key. The normal values are obtained by PEEKing the table values used before POKEing in your values. With the RESTORE, POKEing in the normal values after the new values are in will not reactivate the RESTORE key.

The STOP key is easy to do away with. If you're using the VIC for a demonstration, it's nice to have it unstoppable.

LISTing is a bit more complicated to disable. One way to do it is to embed a control character that will cause the listed characters to be the same color as the screen. For example, white letters on a white screen result in a blank screen. The table gives the method. Put a REM line in the program and follow the instructions. Everything after the REM " will be invisible.

The super POKE is POKE 808,100. STOP, RESTORE, and LIST are now all dead. LIST isn't fully dead, but it might as well be since it prints only nonsense.

Before you use any of these commands, make sure that your program is running the way you want it to. Now add the lines that you want from the table. Usually, adding them as the first line of the program is best. Do *not* RUN the program yet, however. Double-check to make sure all the numbers are right, and SAVE the program. The table POKEs won't take effect until the program is RUN. The program is now safe on tape. When it is LOADed and RUN, the POKEs will take effect.

**VIC Memory Addresses In RAM**
*(Some 64 addresses also given)*

| Function | Protection Method | Normal Values |
|---|---|---|
| STOP, RESTORE, and LIST | (VIC) POKE 808,100<br>(64) POKE 808,225 | (VIC) POKE 808,112<br>(64) POKE 808,237 |
| STOP | (VIC) POKE 808,127<br>(64) POKE 788,52 | (VIC) POKE 808,112<br>(64) POKE 788,49 |
| RESTORE | POKE 792,90<br>POKE 793,203 | —— |
| SAVE | POKE 818,73<br>POKE 819,245 | POKE 818,113<br>POKE 819,246 |
| LIST (VIC or 64) | 1. REM " "<br>2. press CONTROL and 9<br>3. move cursor to second quote mark<br>4. press SHIFT and INST DEL<br>5. press SHIFT and M<br>6. press SHIFT and INST DEL<br>7. press CONTROL and screen color<br>8. press RETURN | |

©

# USR Sort

Walter D. Thompson, Jr.

*One way to create your own operands (commands) is to write subroutines that can be called from a BASIC program. You can, for example, use the Atari's USR function to create a routine with the speed of machine language. The time-saving routine can be especially important in sorting programs.*

How many times have you created a simple file to which you have made several additions? Later, you discover that you want to list the records in the file in some order other than the sequentially entered order. Or perhaps you would even like to rewrite the file in a new order.

A sort utility in DOS could very well handle all possible demands. However, a simple array sort can handle many smaller applications by reading the fields or records into an array in memory and sorting the array. The only limitation is available memory.

## The Sort Routine

A sorting routine is a method by which related elements are compared and ranked. In the Atari, this ranking is accomplished by comparing the ATASCII values. Thus, if we had an array with 26 elements, each one byte long, we could load the array with each letter of the alphabet, and after sorting the array, we would have the letters in order from A to Z.

The sort routine that we want would need to compare a string of letters to another string, perhaps of unequal length. The sort order is left justified so that an element beginning with a D would come before an element starting with an N. However, we also want DUSTY to come after DUST and before DUTY.

Our sort routine must compare each element byte to the corresponding byte of the elements before and after.

Some languages have built-in operands which accomplish this in some manner. For example, IBM BASIC operands AIDX and DIDX return a second array which has sorted pointers to the first array.

## Atari Restrictions And Solutions

When Atari BASIC was written, several operands were omitted simply because there was not sufficient internal memory for all of the special operands available. However, BASIC does allow the user to build assembler subroutines, which can be called from the BASIC program. The utility of the Atari is greatly enhanced by the user's ability to write many special applications. These applications can then use the far greater speed of machine language.

Speed is particularly important in sort functions. Anyone who has written a BASIC program to do a bubble sort on a string array, and has then waited for many minutes for strings of moderate length to process, can attest to the importance of speed in a sort. A string of file-size proportions could very well take hours. This is because a sort, of any type, requires many repetitive compares to accomplish its final result. These are time-consuming operations in BASIC, but they are extremely fast with machine language.

## Locating The Sort

So it seems that we must simply load a machine language sort module and call it from a BASIC program. But first we must consider where to locate the machine language module. It could be in the same place for any program which would load and access it, yet it must not interfere with the operating system or with our BASIC program which has called it. We would not accomplish very much if the sort module were located in the very middle of the array to be sorted.

The first sort utility I wrote worked extremely well and fit entirely into page six. So what more could I want from a sort that handled a large (32,000) element array and executed in mere seconds?

The answer is *safety*. Upon closer examination, I realized that a simple error in programming could result in the programmer sorting all of his or her program and memory above the array. This could include the display list, screen memory, and even BASIC itself. The routine needed a method of validating the total length of the sort array. We could have in memory an array of 100 elements of 127 bytes each, but through error, pass a value of 1000 elements to the sort routine. And the sort routine would, of course, attempt to sort 124K of memory, being forced to stop only when it attempted to sort ROM.

After developing and adding an error-check to make the program safe, I discovered that my original memory location was no longer able to hold the improved version. After some more juggling, I decided the solution was to make the object code relocatable. This means that the program permitted no jumps to a specified address; it allowed only relative branches.

Although the sort routine is relocatable, it does make use of a number of page zero and page six memory locations when executed. During the course of a sort, the program will alter the contents of locations $CB-$D1 and $6F8-$6FF.

## Building The Array

We want to sort a group of unequal fields in a string array of as yet unspecified size. We must work with elements of a specified length, padding with blanks to the right for alphanumeric fields and zeros to the left for numeric fields. For example, a name field to be sorted may have a maximum length of 25 characters. There are 100 names or elements to be sorted. And so:

```
LET MLENGTH=25
LET ELEMENTS=100
LET X=MLENGTH*ELEMENTS
DIM SARRAY$(X)
```

In order to load the array so that the sort routine has equivalent elements to compare, we assign each field the maximum length using the following technique:

```
DIM FIELD$(MLENGTH)
  FOR I=1 TO MLENGTH:LET FIELD$(I)=" ":
    NEXT I
  LET L=LEN(NAME$)
  LET FIELD$(1,L)=NAME$
  LET P=ELEMENT*MLENGTH
  LET SARRAY$(P,P+MLENGTH-1)=FIELD$
```

If numeric fields are to be loaded into a string array for sorting, the following technique will build equal length elements with leading zeros:

```
DIM FIELD$(MLENGTH),TEST$(MLENGTH)
  FOR I=1 TO MLENGTH:LET FIELD$(I)="0":
    NEXT I
  LET TEST$=STR$(ZIPCODE)
  LET L=LEN(TEST$)
  LET FIELD$(MLENGTH+1-L,MLENGTH)=
    TEST$
```

We have now built an array for sorting, retaining the number of elements and the length of each element.

## The USR Call

Upon calling the sort routine, we specify the number of elements to be sorted, the length of each element, the sort order, and the relative variable number of our array.

We call the sort routine by executing the following USR call:

```
LET EFLAG=USR(STARTADDR,ELEMENTS,
  MLENGTH,ORDER,RELVARNUM)
```

As long as the program does not change graphics modes, at least prior to all sorting, a very handy place to locate the sort routine object code is just below RAMTOP. The STARTADDR= PEEK(741)+256*PEEK(742) -339. Or in BASIC A+, STARTADDR=DPEEK(741) -339. ELEMENTS should be the actual number of elements loaded into the array. The additional routine which validates the length of the array is based on the LEN value of the variable, not its DIMed length. This helps to prevent garbage from being mixed in accidentally with good data. In our case, MLENGTH is 25, although the range of values for length is from 1 to 255 bytes. ORDER determines whether we will sort the array elements in ascending or descending order; 1 represents ascending order, and 255 indicates descending order. RELVARNUM, or the relative variable number, is the relative appearance of our string array in the BASIC program. Looking back to our first example, we see that MLENGTH was the first variable in our example, ELEMENTS was second, X third, and SARRAY$ fourth. The first variable has a relative number of zero, so RELVARNUM=4 -1 or RELVARNUM=3.

When a USR routine is called, a value may be returned to the BASIC program. This value is used to return error conditions. Those error conditions are represented by values of EFLAG, which are described as follows:

| A Value of | Indicates |
|---|---|
| 0 | An error-free sort |
| 1 | ELEMENTS*MLENGTH > ARRAY LENGTH |
| 2 | Sort ORDER <> 1 or 255 |
| 3 | MLENGTH=0 or >255 |
| 4 | ELEMENTS=0 |
| 5 | Variable not a DIMed string array |
| or | relative variable number exceeds 31 |
| 6 | Invalid number of parameters passed |

## The Sort Utility

Our sort utility works by comparisons made in a loop, from the most significant byte to the least significant byte. Comparisons are made until two compared bytes are found out of order or until the element is fully tested. The order of comparison reduces the number of passes through the loop to a minimum.

To further increase efficiency, only one pass is made through the array. When one element is exchanged with a lower one, it is tested against the next lower element, until it is no longer exchanged or until it is placed at the bottom of the array. After placing that element, the routine returns to the element's original position and moves to the next element. In this way, each element is ranked among all previously sorted elements.

Of course this is not a DOS utility sort, and it is limited by available memory. However, the USR function allows us to use machine language speeds in our routine. And with a little programming and thought, most files can be sorted on any field by placing the sort field first in the element string, followed by the entire record.

For convenience, the machine language sort routine is provided as a BASIC loader program. For anyone who would prefer a copy on diskette or tape, please send a self-addressed, stamped mailer, a blank tape or diskette, and $3 to:

*Walter D. Thompson, Jr.*
*P.O. Box 6602*
*Greensboro, NC 27405*

All diskettes will be written in DOS 2.0 format. In addition, all copies of the BASIC program listed here will be accompanied by an Assembler version and the original version of the object code which fits in page six.

### USR Sort

```
290 PRINT "{CLEAR}":PRINT "ARRAY SOR
    T PROGRAM":PRINT
300 TRAP 300:PRINT "INPUT # OF ELEME
    NTS TO SORT";:INPUT ELEMENTS
310 TRAP 310:PRINT "INPUT MAXIMUM EL
    EMENT LENGTH";:INPUT LENGTH
320 TRAP 320:PRINT "INPUT SORT ORDER
     - ENTER 1 OR 255";:INPUT ORDER
330 PRINT :PRINT "INITIALIZING..."
340 DIM ARRAY$(LENGTH*ELEMENTS),ELEM
    ENT$(LENGTH),BLANK$(LENGTH),WORK
    $(LENGTH):GOSUB 700
350 PRINT "{CLEAR}":REM ****  CLEAR
    SCREEN
360 PRINT "ARRAY SORT ENTRY":PRINT
370 VNUM=3:REM ****  ARRAY$ IS THE 4
    TH VARIABLE TO APPEAR IN PROGRAM
380 FOR X=1 TO LENGTH:BLANK$(X)=" ":
    NEXT X
390 FOR I=0 TO ELEMENTS-1
400 WORK$=BLANK$
410 TRAP 410:PRINT "INPUT UP TO ";LE
    NGTH;" CHARACTERS":INPUT ELEMENT
    $
420 L=LEN(ELEMENT$):IF L=0 THEN 450
430 S=LENGTH*I+1:E=LENGTH*(I+1):WORK
    $(1,L)=ELEMENT$:ARRAY$(S,E)=WORK
    $:NEXT I
440 I=ELEMENTS:REM ***  ESTABLISH I
    IF ALL ELEMENTS FILLED
450 EFLAG=USR(START,I,LENGTH,ORDER,V
    NUM)
460 ON EFLAG GOTO 470,480,490,500,51
    0,520:GOTO 530
470 PRINT "**ERROR S/";EFLAG;" - ARR
    AY LENGTH EXCEEDED":STOP
480 PRINT "**ERROR S/";EFLAG;" - INV
    ALID SORT ORDER":STOP
490 PRINT "**ERROR S/";EFLAG;" - ELE
    MENT LENGTH INVALID":STOP
500 PRINT "**ERROR S/";EFLAG;" - ELE
    MENT COUNT = 0":STOP
510 PRINT "**ERROR S/";EFLAG;" - INV
    ALID VARIABLE TYPE":STOP
520 PRINT "**ERROR S/";EFLAG;" - INV
    ALID # PARAMETERS":STOP
530 FOR J=0 TO I-1
540 S=LENGTH*J+1:E=LENGTH*(J+1):PRIN
    T ARRAY$(S,E):NEXT J
550 STOP
700 REM **  LOCATE OBJECT BELOW OS M
    EMTOP
710 START=PEEK(741)+256*PEEK(742)-33
    9
720 TRAP 725:FOR I=0 TO 338:READ A:P
    OKE (START+I),A:NEXT I
730 RETURN
740 PRINT "ERROR DURING OBJECT LOAD"
    :STOP
750 DATA 169, 0, 133, 212, 133, 213,
     104, 168
760 DATA 240, 109, 201, 4, 240, 7, 1
    04, 104
770 DATA 136, 208, 251, 240, 98, 104
    , 141, 251
780 DATA 6, 133, 207, 104, 141, 250,
     6, 133
790 DATA 206, 104, 240, 4, 169, 1, 1
    33, 213
800 DATA 104, 133, 205, 133, 203, 10
    4, 240, 4
810 DATA 169, 2, 133, 213, 104, 133,
     204, 104
820 DATA 240, 4, 169, 4, 133, 213, 1
    04, 10
830 DATA 176, 55, 10, 176, 52, 10, 1
    76, 49
840 DATA 168, 165, 213, 240, 8, 201,
     2, 48
850 DATA 44, 240, 44, 16, 36, 177, 1
    34, 201
860 DATA 129, 208, 30, 165, 140, 200
    , 200, 24
870 DATA 113, 134, 141, 252, 6, 141,
     254, 6
880 DATA 165, 141, 105, 0, 200, 113,
     134, 141
890 DATA 253, 6, 141, 255, 6, 208, 1
    7, 230
900 DATA 212, 230, 212, 230, 212, 23
    0, 212, 230
910 DATA 212, 230, 212, 169, 0, 133,
     213, 96
920 DATA 200, 200, 169, 0, 24, 141,
    249, 6
930 DATA 141, 248, 6, 162, 8, 102, 2
    03, 144
```

```
940 DATA 26, 24, 173, 248, 6, 101, 2
    Ø6, 141
950 DATA 248, 6, 173, 249, 6, 101, 2
    Ø7, 141
960 DATA 249, 6, 176, 213, 165, 205,
     56, 229
970 DATA 203, 144, 206, 6, 206, 38,
    207, 202
980 DATA 208, 219, 177, 134, 205, 24
    9, 6, 48
990 DATA 192, 208, 8, 136, 177, 134,
     205, 248
1ØØØ DATA 6, 48, 182, 165, 205, 24Ø,
      174, 165
1Ø1Ø DATA 204, 201, 1, 24Ø, 4, 201,
     255, 208
1Ø2Ø DATA 166, 173, 25Ø, 6, 13, 251,
      6, 24Ø
1Ø3Ø DATA 154, 173, 254, 6, 133, 208
     , 24, 1Ø1
1Ø4Ø DATA 2Ø5, 141, 254, 6, 173, 255
     , 6, 133
1Ø5Ø DATA 2Ø9, 1Ø5, Ø, 141, 255, 6,
     2Ø6, 25Ø
1Ø6Ø DATA 6, 2Ø8, 6, 2Ø6, 251, 6, 16
     , 1
1Ø7Ø DATA 96, 16Ø, Ø, 24, 165, 2Ø8,
     1Ø1, 2Ø5
1Ø8Ø DATA 133, 2Ø6, 165, 2Ø9, 1Ø5, Ø
     , 133, 2Ø7
1Ø9Ø DATA 177, 2Ø6, 2Ø9, 2Ø8, 24Ø, 8
     , 8, 1Ø4
11ØØ DATA 69, 2Ø4, 48, 9, 16, 195, 2
     ØØ, 196
111Ø DATA 2Ø5, 24Ø, 19Ø, 2Ø8, 235, 1
     6Ø, Ø, 177
112Ø DATA 2Ø6, 17Ø, 177, 2Ø8, 145, 2
     Ø6, 138, 145
113Ø DATA 2Ø8, 2ØØ, 196, 2Ø5, 2Ø8, 2
     41, 165, 2Ø8
114Ø DATA 56, 229, 2Ø5, 176, 2, 198,
      2Ø9, 133
115Ø DATA 2Ø8, 165, 2Ø9, 2Ø5, 253, 6
     , 48, 153
116Ø DATA 2Ø8, 183, 165, 2Ø8, 2Ø5, 2
     52, 6, 16
117Ø DATA 176, 48, 142
```

©

# Working With SID

Jerry M. Jaco

*In this unique approach to the Commodore 64's SID chip, the author discusses the SID chip's anatomy and capabilities in the context of its essential similarity to the design of music synthesizers.*

---

If you've decided you want to make music on your Commodore 64, and you've read all the literature on the subject and still don't know where to begin, perhaps a look at how an analog synthesizer is used in an electronic music studio will clarify many aspects of the 64's amazing sound capabilities. Once we have covered the physical aspects of a synthesizer, we can begin to understand some of the techniques used to create sounds artificially.

Electronic music studios usually include at least one analog synthesizer. Most synthesizers have a modular design which allows the synthesizer to be built and expanded according to the dictates of budget, space, and ability. Each module on the synthesizer has a different function, and the builder-user is free to duplicate or omit these functions in any way he sees fit.

Each module on the synthesizer is independent of all the others. The only way to connect them is either by a panel of fancy selector switches or via the more common *patch cords*. Patch cords are simply pieces of electrical cable of varying lengths which have standard plugs attached on each end. Plugging one end of a patch cord into the output socket of one module and the other end of the patch cord into the input socket of another module creates an electrical pathway called a *patch*.

If a patch leads from a Source module, such as an oscillator, to an Output module, such as a mixer, the resulting sound will be audible to the outside world. (See Figure 1.) The term *signal* is used to describe the electrical current being passed from one module to another. A *source signal* is one that will eventually be heard as a real sound. A *control signal* is a varying voltage used to electronically control another module. It does not contain sound information per se.

## A "Patch" For The SID Chip

In Figure 1, there is only one source signal being processed by the mixer. A mixer can handle up to three source signals on our hypothetical system, which it combines into one composite signal that gets sent on to the speakers, and to your ears. (See Figure 2.) On the 64, Program 1 accomplishes exactly what the analog synthesizer does in Figure 2.

### Program 1: Three Voices – Or A Chord

```
10 FORI=0TO24:POKE54272+I,0:NEXT
20 POKE54272,37:POKE54273,17:REM OSC1

30 POKE54279,229:POKE54280,22:REM OSC2

40 POKE54286,214:POKE54287,28:REM OSC3

50 POKE54276,17:POKE54283,17:POKE54290,17
   : REM TRIANGLE WAVE FOR ALL OSC'S

60 POKE54278,245:POKE54285,245:POKE54292,
   245:REM SUS/REL VALUES FOR ALL OSC'S

70 POKE54296,15:REM MASTER VOLUME ON

75 FORT=1TO500:NEXT:REM CHORD DURATION

80 POKE54276,16:POKE54283,16:POKE54290,16

90 FORT=1TO250:NEXT:REM RELEASE DURATION

95 POKE54296,0:REM TURN OFF VOLUME:rem 29
96 END
```

This is a very basic "patch" for the Sound Interface Device (SID) chip on the 64. Lines 20, 30, and 40 set the frequencies of the three oscillators. Line 20 POKEs the values for middle C into Voice 1. Line 30 POKEs the values for F into Voice 2, and line 40 POKEs the values for A into Voice 3. This gives us a "chord," which is simply three notes (voices) sounding simultaneously. Line 50 selects a triangle wave output for all three voices. Line 70 is the mixer volume control. When the value 15 is POKEd into this location, the master volume control is turned all the way up. When 0 is POKEd, the volume control is turned off, as in line 55. The other lines will become clearer as we go along.

**Figure 1:** Processing A Single Source Signal

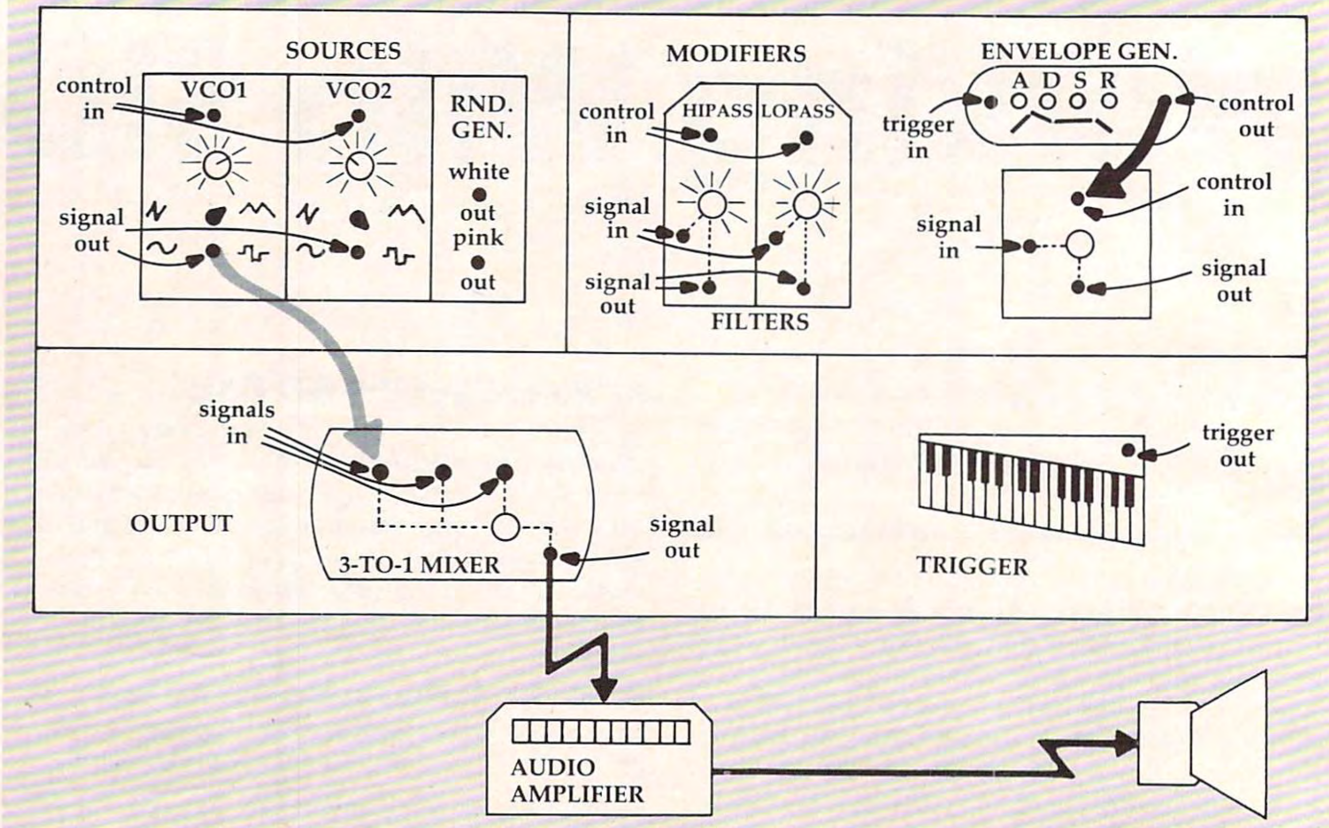
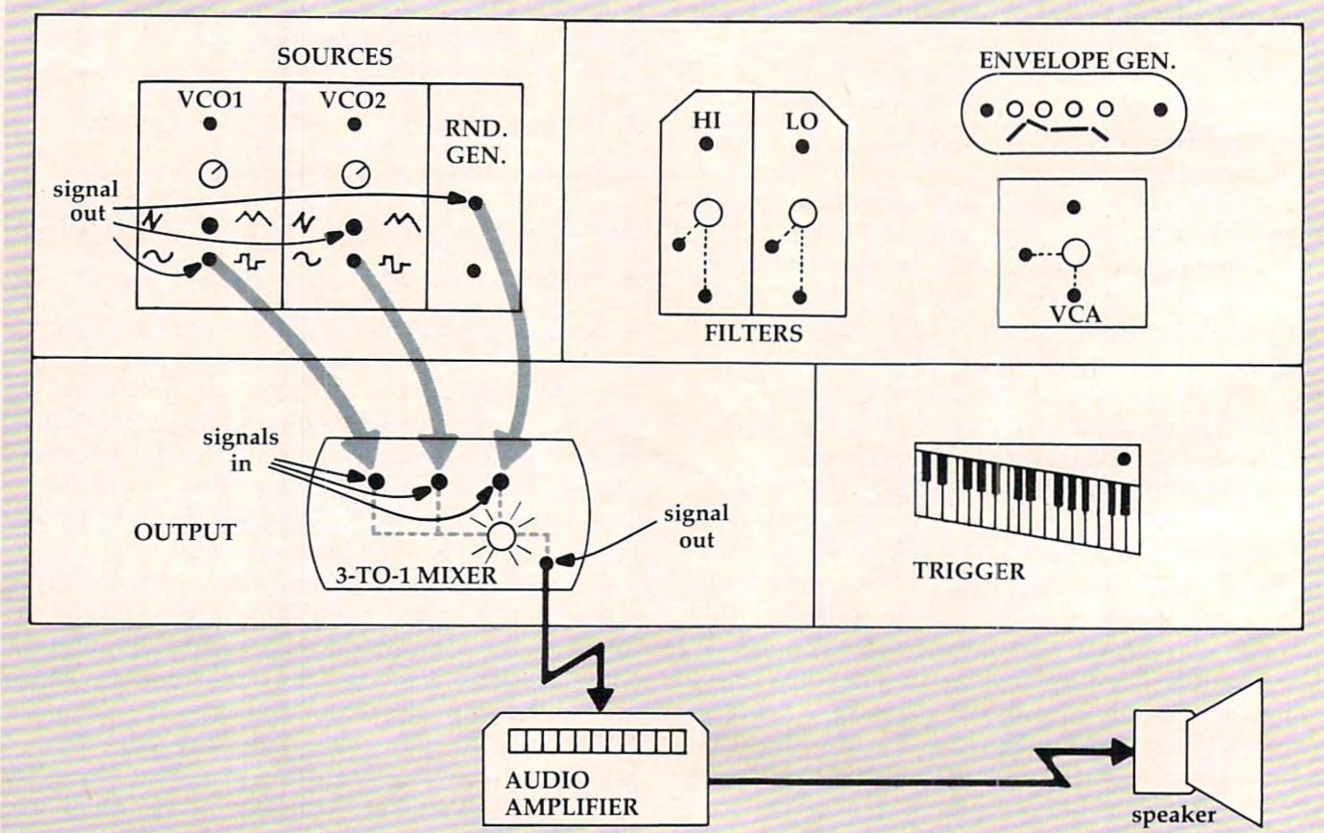



**Figure 2:** Processing Three Source Signals



# SANYO MONITOR SALE!!


9" Data Monitor

- 80 Columns × 24 lines
- Green text display
- East to read - no eye strain
- Up front brightness control
- High resolution graphics
- Quick start - no preheating
- Regulated power supply
- Attractive metal cabinet
- UL and FCC approved

- *15 Day Free Trial - 90 Day Immediate Replacement Warranty*

| | |
|---|---|
| 9" Screen - Green Text Display | $ 79.00 |
| 12" Screen - Green Text Display (anti-reflective screen) | $ 99.00 |
| 12" Screen - Amber Text Display (anti-reflective screen) | $119.00 |

***Display Monitors From Sanyo***

With the need for computing power growing every day, Sanyo has stepped in to meet the demand with a whole new line of low cost, high quality data monitors. Designed for commercial and personal computer use. All models come with an array of features, including upfront brightness and contrast controls. The capacity 5 × 7 dot characters as the input is 24 lines of characters with up to 80 characters per line.

Equally important, all are built with Sanyo's commitment to technological excellence. In the world of Audio/Video, Sanyo is synonymous with reliability and performance. And Sanyo quality is reflected in our reputation. Unlike some suppliers, Sanyo designs, manufactures and tests virtually all the parts that go into our products, from cameras to stereos. That's an assurance not everybody can give you!

SANYO

*Official Video Products of the Los Angeles 1984 Olympics*

**• LOWEST PRICES • 15 DAY FREE TRIAL • 90 DAY FREE REPLACEMENT WARRANTY**
**• BEST SERVICE IN U.S.A. • ONE DAY EXPRESS MAIL • OVER 500 PROGRAMS • FREE CATALOGS**

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

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

**PROTECTO**

**ENTERPRIZES** (WE LOVE OUR CUSTOMERS)

**BOX 550, BARRINGTON, ILLINOIS 60010**
**Phone 312/382-5244 to order**

On an analog synthesizer, *pots* (potentiometers) are controls that do things such as raise and lower the volume of a sound signal or change the frequency (pitch) of an oscillator. Pots are also the main components of game paddles and TV volume controls. To make new sounds on an analog synthesizer, the user will twist pots on each module and listen for the resulting effect. When he finds one he likes, he either records the sound on tape or writes down the patch on a patch chart, marking the pathways made by the patch cords and the positions of the pots for future reference. Analog synthesizers are very useful in this way because drastic changes in a sound can be quickly made by simply twisting a knob or plugging a patch cord into something else.

## Turning Knobs With POKEs

A digitally-controlled synthesizer, such as our SID chip, uses numbers POKEd into control registers to accomplish the same things that knob-twisting and patch-cord-plugging do on an analog synthesizer. For example, if you POKE a 16-bit value into the first two registers of the SID chip (54272 and 54273), you've set the frequency value for Oscillator 1. POKE a four-bit number into the high nybble of the sixth register on the chip, and you've set the Attack value of the envelope for Oscillator 1. POKEing different values into other registers will activate them in the same way that turning the pots or setting switches will activate the analog synthesizer modules.

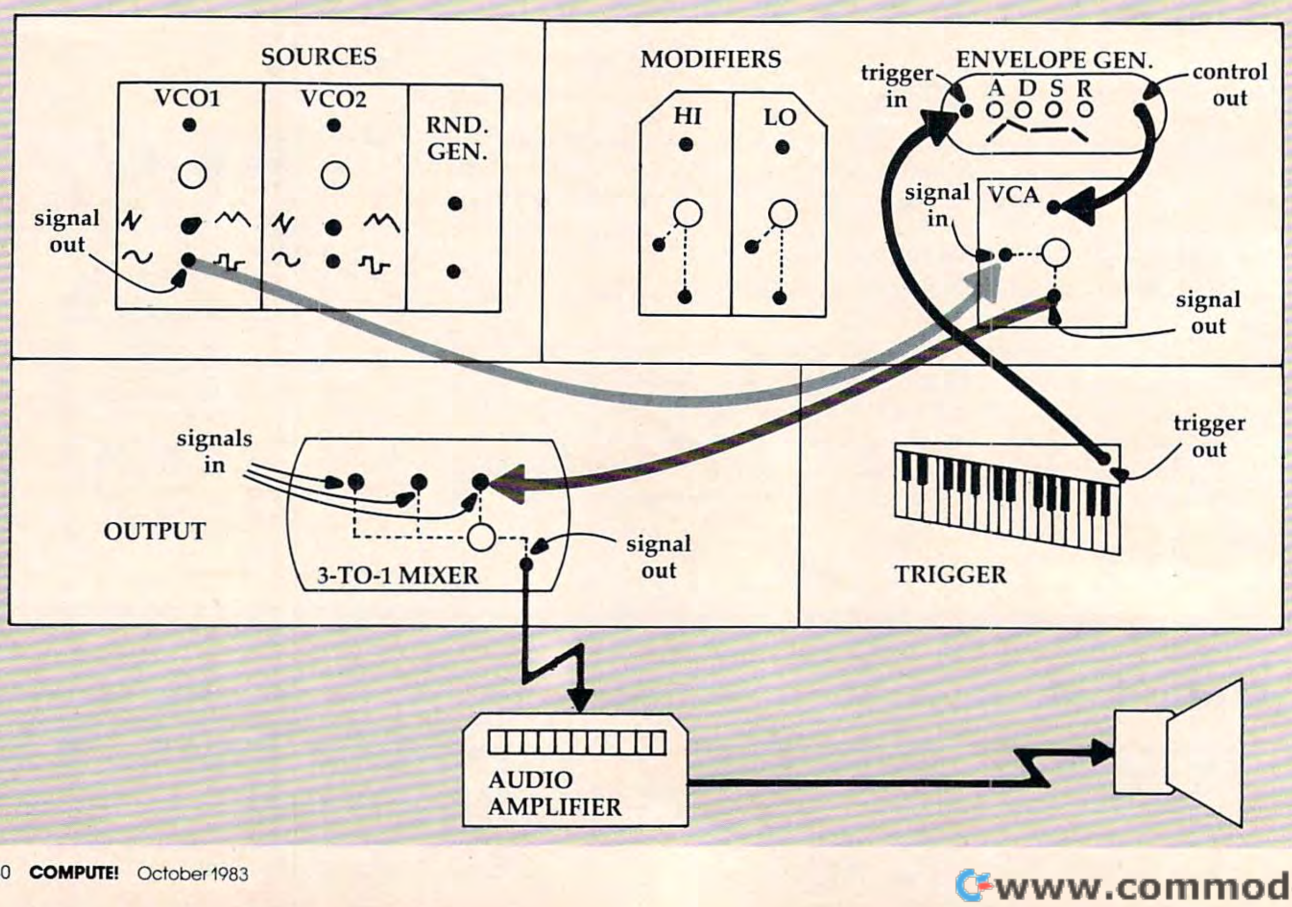
## Envelope Generation

Look at Figure 1 again. It shows a direct path from an oscillator (VCO1) to the mixer. If we were to break that path, sending the output of VCO1 to the input of the amplifier module (VCA), we would then need to send the output of VCA to the mixer so that the sound from VC01 could still be heard. The patch shown in Figure 3 would be the result. Now we can make VCO1's signal even louder by adjusting the pot on VCA or on the mixer. The real reason for taking this route is that the envelope generator can be brought into play, since it directly controls the VCA.

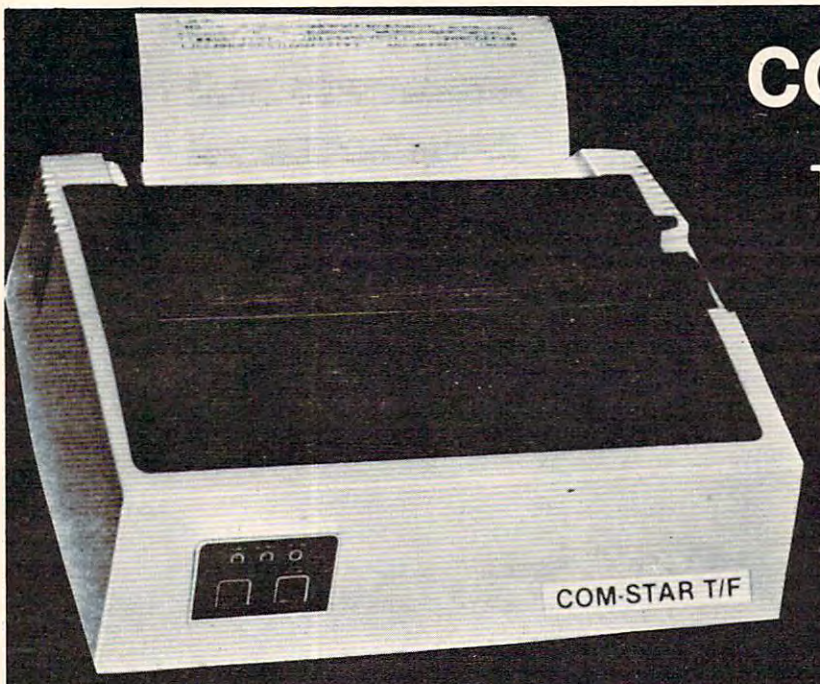
There are four pots on the envelope generator module. The first controls the Attack time; the second, the Decay time. The third sets the Sustain level, and the fourth controls the Release time. On the SID, two registers in high-low nybble format control these functions. The most important function is perhaps the Sustain level. It is not a timing value, but is rather the level at which the amplifier's volume control is set while the note is being sounded. If the Sustain level is zero, no sound will be heard after the Attack and Decay phases have ended.

The envelope generator puts out an electrical signal which tells the amplifier when to turn up

**Figure 3: Using The VCA**



# 80 COLUMN PRINTER SALE—$149.00*




## COM-STAR T/F

**Tractor Friction Printer**

**only $219**

(Parallel or Serial)

• Lowest price quality tractor friction printer in the U.S.A. • Fast 80 characters per second • 40, 46, 66, 80, 96, or 132 characters per line spacing • Prints labels, letters, graphs, and tables • List your programs • Print out data from modem services

### *STX-80 COLUMN PRINTER—$149.00

Prints full 80 columns. Super silent operation, 60 CPS, prints Hi-resolution graphics and block graphics, expanded character set, exceptionally clear characters, fantastic print quality, uses inexpensive thermal roll paper!

### DELUXE COMSTAR T/F PRINTER—$219.00

The Comstar T/F is an excellent addition to any micro-computer system. (Interfaces are available for Apple, VIC-20, Commodore-64, Pet, Atari 400 and 800, and Hewlett Packard). At only $219 the Comstar gives you print quality and features found only on printers costing twice as much. Compare these features.

- **BI-DIRECTIONAL PRINTING** with a **LOGIC SEEKING CARRIAGE CONTROL** for higher through-put in actual text printing. 80 characters per second.
- **PRINTING VERSATILITY:** standard 96 ASCII character set plus block graphics and international scripts. An EPROM character generator includes up to 224 characters.
- **INTERFACE FLEXIBILITY:** Centronics is standard. Options include EIA RS232C, 20mA Current Loop.
- **LONG LIFE PRINT HEAD:** 100 million character life expectancy.
- **THREE SELECTABLE LINE SPACINGS:** 6, 8 or 12 lines per inch.
- **THREE SELECTABLE CHARACTER PITCHES:** • 10, 12 or 16.5 characters per inch. 132 columns maximum. Double-width font also is standard for each character pitch.
- **PROGRAMMABLE LINE FEED:** programmable length from 1/144 to 255/144 inches.
- **VERTICAL FORMAT CONTROL:** programmable form length up to 127 lines, useful for short or over-sized preprinted forms.
- **FRICTION AND TRACTOR FEED:** will accept single sheet paper.
- **224 TOTAL CHARACTERS**
- **USES STANDARD SIZE PAPER**

If you want more try —

### Premium Quality COMSTAR T/F SUPER-10X PRINTER—$299.00
### More Features Than RX-80

For $299 you get all of the features of the Comstar T/F plus 10" carriage 120 cps, 9 x 9 dot matrix with double strike capability for 18 x 18 dot matrix. High resolution bit image (120 x 144 dot matrix), underlining, backspacing, left and right margin settings, true lower descenders, with super and subscripts, and prints standard, Italic, Block Graphics, special characters, plus 2K of user definable characters. For the ultimate in price performance the Comstar T/F Super 10" leads the pack!

### Double Immediate Replacement Warranty

We have doubled the normal 90 day warranty to 180 days. Therefore if your printer fails within "180 days" from the date of purchase you simply send your printer to us via United Parcel Service, prepaid. We will IMMEDIATELY send you a replacement printer at no charge via United Parcel Service, prepaid. This warranty, once again, proves that WE LOVE OUR CUSTOMERS!

## 15 DAY FREE TRIAL

### OTHER OPTIONS

| | |
|---|---|
| Extra Ribbons | $ 5.95 |
| Roll Paper Holder | 32.95 |
| Roll Paper | 4.95 |
| 5000 Labels | 19.95 |
| 1100 Sheets Fan Fold Paper | 13.95 |

Add $17.50 shipping, handling and insurance. Illinois residents please add 6% tax. Add $40.00 for CANADA, PUERTO RICO, HAWAII, ALASKA orders. WE DO NOT EXPORT TO OTHER COUNTRIES. Enclose cashiers check, money order or personal check. Allow 14 days for delivery, 2 to 7 days for phone orders. 1 day express mail available!! Canada orders must be in U.S. dollars.

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

SUPER-10" ABCDEFGHIJKLMNOPQRSTUVWXYZ
ABCDEFGHIJKLMNOPQRSTUVWXYZ 1234567890

GRAFSTAR (Apple interface card - "Like Grappler Plus") SALE $69.00.

the volume and how long it should take, as well as how high to set the volume, and when and how long to turn it all the way off again. This is why the amplifier module in the diagrams is called "VCA." This stands for Voltage Controlled Amplifier and means that the amplifier can be controlled by an incoming variable voltage, such as the one supplied by the envelope generator.

## ADSR Values

On the SID chip, each voice has its own envelope generator. Within the group of seven registers (0-6) that control the three oscillators, register 5 contains the Attack and Decay values in high-low nybble format, and register 6 contains the Sustain/Release values. All values are four-bit numbers (nybbles). The Attack value determines how long the amplifier should take to reach peak amplitude (maximum volume).

The Decay value determines how long the amplifier should take to go from peak amplitude to the level specified by the Sustain value. The Release value is the time the amplifier will use to return to the lowest amplitude level ("off") from the Sustain level.

Remember, though, that on the analog synthesizer as well as on the SID chip, the envelope will not go into effect until it is "triggered." The lowest order bit (bit 0), the Gate bit, triggers each envelope on the SID chip. On the analog synthesizer, triggering of the envelope is accomplished through the use of an attached keyboard module. When a key is pushed down (and as long as it is held down), the Attack, Decay, and Sustain values will go into effect in order. When the key is released, the Release phase is triggered, and the VCA will close down the volume of the signal it is operating on over the length of time specified by the Release value.

Program 2 demonstrates the effect of the various ADSR values:

## Program 2: Effects Of The ADSR Values

```
100 FORI=0TO24:POKE54272+I,0:NEXT
110 POKE54272,37:POKE54273,17:REM OSC1

120 POKE54276,129:REM NOISE WAVE OSC 1

130 POKE54277,240:REM SLOW ATTACK/FASTEST
    DECAY RATE
140 POKE54278,240:REM HIGHEST SUSTAIN LEV
    EL/FASTEST RELEASE RATE
150 POKE54296,15:REM FULL VOL. AT MIXER

160 FORT=1TO4500:NEXT:REM DURATION FOR AT
    TACK,DECAY, AND SUSTAIN
170 POKE54276,128:REM BEGIN RELEASE CYCLE

180 FORT=1TO4500:NEXT:REM REL. DURATION

190 POKE54296,0:REM TURN OFF VOLUME

191 END
```

In line 130, the Attack value is all the way on, and the Decay value is all the way off. In line 140, the Sustain value is all the way on and the Release value is off. Each value is a four-bit number, 0 to 15. With the Attack and Sustain setting, the actual POKE value is shifted to the high nybble; thus, 240 is actually the Attack value equal to 15 (for slowest Attack) multiplied by 16. The sound generated is a random noise that gradually gets louder and then stops suddenly. It stops suddenly because we have set the Release value to 0, allowing no time for a gradual decrease in volume.

Change the value 240 in line 140 to 255 and RUN the program again. The sound should slowly fade away. The high nybble of 54278 (Sustain) is now 240 and the low nybble (Release) 15, making a total of 255, the value we just POKEd into 54278. Try lowering the Sustain value by two or three (2*16 or 3*16); that is, POKE 54278 with either 223 or 207 and see what happens. The sound should build up as before but should then fall off markedly. Change the Decay value from 0 in line 130 to about 8 (POKE 54277,248) and hear how the drop-off is now smoothed out. Similarly, shorten the Attack time to vary the start of the sound the same way the Sustain value was altered. The results should be vastly different from those we started with, and we've been working with only two registers!

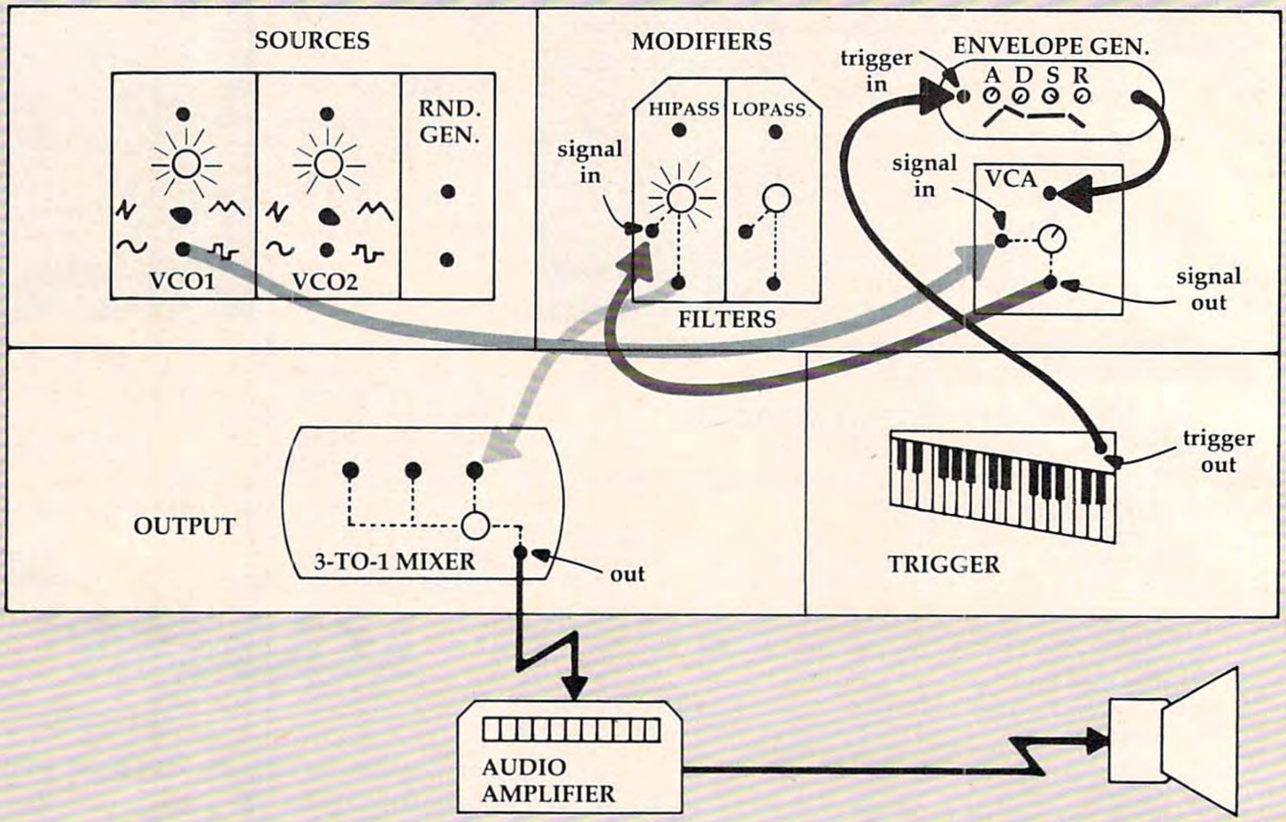
Look now at line 170. Notice that we subtracted one from the value we originally POKEd into 54276 in line 120. This zeros the Gate bit in 54276, and it is the same as taking your finger off the keyboard on the analog synthesizer: the Release cycle gets triggered. Of course, it works only if the VCA Sustain level has been previously raised high enough to hear the tone. The delay loop in line 180 is also necessary to allow the Release cycle to reach its lowest level.

## Using Filters To Color Sound

Let's add a filter to the path in Figure 3. The path from the VCA to the mixer is broken so that filtering the modulated signal will be more easily heard. In our diagram, we have a choice of a high-pass or low-pass filter. On the SID chip, we can also utilize a Band-Pass filter.

The pot on each filter is used to adjust the cutoff frequency, which is the frequency above which a high-pass filter allows frequencies in the sound spectrum to be heard and below which the filter suppresses them. The low-pass filter is the opposite of the high-pass filter in that it suppresses the frequencies above the cutoff value and allows those below it to sound. A Band-Pass filter allows frequencies to be heard within a narrow band surrounding the cutoff frequency (called a center frequency in this case), while suppressing all the rest. Use of filters constitutes a technique called

**Figure 4:** Using A High-Pass Filter




*subtractive synthesis*, which selectively eliminates available frequencies of the sound spectrum, producing widely varying sound colors.

Figure 4 indicates that we've decided to filter VCO1 through a high-pass filter. VCO1 is set to produce a sawtooth wave. The path of the patch runs out of VCO1 into the VCA, and from the VCA into the HIPASS filter. From there, the signal heads to the mixer and out to the speaker. Program 3 is a routine that does the same thing.

## Program 3: Filtered Sound

```
200 FORI=0TO24:POKE54272+I,0:NEXT
210 POKE54272,37:POKE54273,17:REM OSC1

220 POKE54276,33:REM SAWTOOTH WAVE OSC1

230 POKE54277,120:REM MED. ATTACK/MED. DE
    CAY
240 POKE54278,245:REM HIGHEST SUSTAIN/MED
    . RELEASE
245 POKE54293,40:POKE54294,5:REM CUTOFF F
    REQUENCY FOR HIGH-PASS FILTER
250 POKE54295,129:REM MED RES'NCE AND OSC
    1 TO BE FILTERED
255 POKE54296,79:REM FULL VOL. AND CHOOSE
     HIGH-PASS FILTER
260 FORJ=1TO250:POKE54294,J:NEXT:REM SWEE
    P CUTOFF FREQ. UPWARDS

270 POKE54276,32:REM BEGIN RELEASE CYCLE

280 FORT=1TO500:NEXT:REM REL. DURATION

290 POKE54296,0:REM TURN OFF VOLUME

295 END
```

To hear the effect of the filter, in line 260 we will sweep the value of the cutoff frequency from low to high. This will allow less and less of the available sound spectrum to be passed by the filter. Listen carefully to the richness of the tone as it is diminished. Switch the wave form to Noise in line 220 by POKEing 129, instead of 33, into 54276 to hear a different version of the effect. Many effects are possible using filters.

## Frequency Modulation

Figure 5 introduces another technique called Frequency Modulation. Notice now that the signal from VCO1 is entering the control input of VCO2, and that the signal from VCO2 is going through the VCA and on to the mixer. The frequency of VCO2 is now being controlled automatically by the output voltage of VCO1 instead of manually by the pot. This is another example of voltage control. The envelope generator controlled the VCA before and an oscillator now controls a VCO (Voltage-Controlled Oscillator).

Frequency Modulation (FM), along with filtering and envelope control, is one of the most significant techniques of sound synthesis. Using one signal source to alter the sound quality of

# EPYX
# TOP 10 GAME SALE
# * BUY TWO — GET ONE FREE *

## SPECIAL FREE GAME
**Buy Two Epyx Games From Protecto And Get A Free Game**

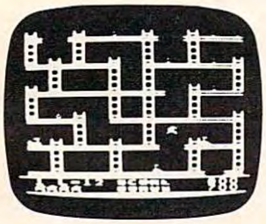
## *The Thinking Man's Paradise*

( Disk / Cassette )

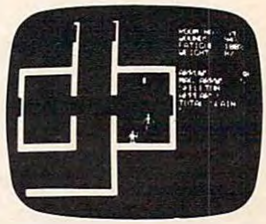
- **Award Winning Games**
- **Fantastic Graphics**
- **Skill (not luck) Needed**
- **Already in top 10 charts**

## COMMODORE - 64 / VIC-20

**Buy Any Two Epyx Games From Protecto And Send The Proof Of Purchase Seals To Epyx And Epyx Will Send You A Free Game.**

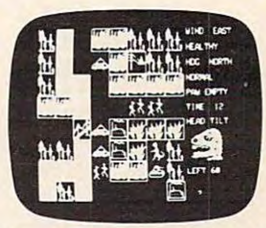


**JUMPMAN**
If you like Donkey Kong, you'll love Jumpman. Over 30 different screens with 8 speeds and 5 skill levels make this the fastest action game in the country. You must leap girders, climb ropes, and scale ladders to reach and diffuse bombs while avoiding robots, birds, bullets, explosives, crumbling walls, vanishing escape routes, and many other obstacles. (Truly a fantastic game!)
List $39.95 **Sale $27.95**

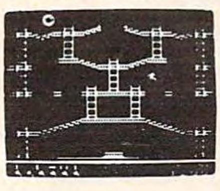


**TEMPLE OF APSHAI (computer game of the year)**
This is the standard by which other adventure games are judged. Full color graphics portray the temple and all its contents - magic, monsters, doomed cities and damsels in distress. Do battle in real time with over 20 monsters, expansion modules will keep your adventure alive in the future.
List $39.95 **Sale $29.95**

**SWORD OF FARGOAL**
Search for the wondrous sword in the depths of an ever changing dungeon. Make yourself invisible, teleport to a new location, drink a healing potion or use enchanted treasures, but watch out for traps and hideous creatures who will try to stop you. (Fantastic dungeon adventure)
List $29.95 **Sale $21.95**



**CRUSH CRUMBLE & CHOMP (Computer game of the year nominee)**
Choose one of six monsters or create your own, and use your monster to destroy one of four unsuspecting cities. The cities aren't totally defenseless, they call on police, national guard, and even a mad scientist, complete with helicopter, to save humanity from the relentless threat.
List $29.95 **Sale $21.95**

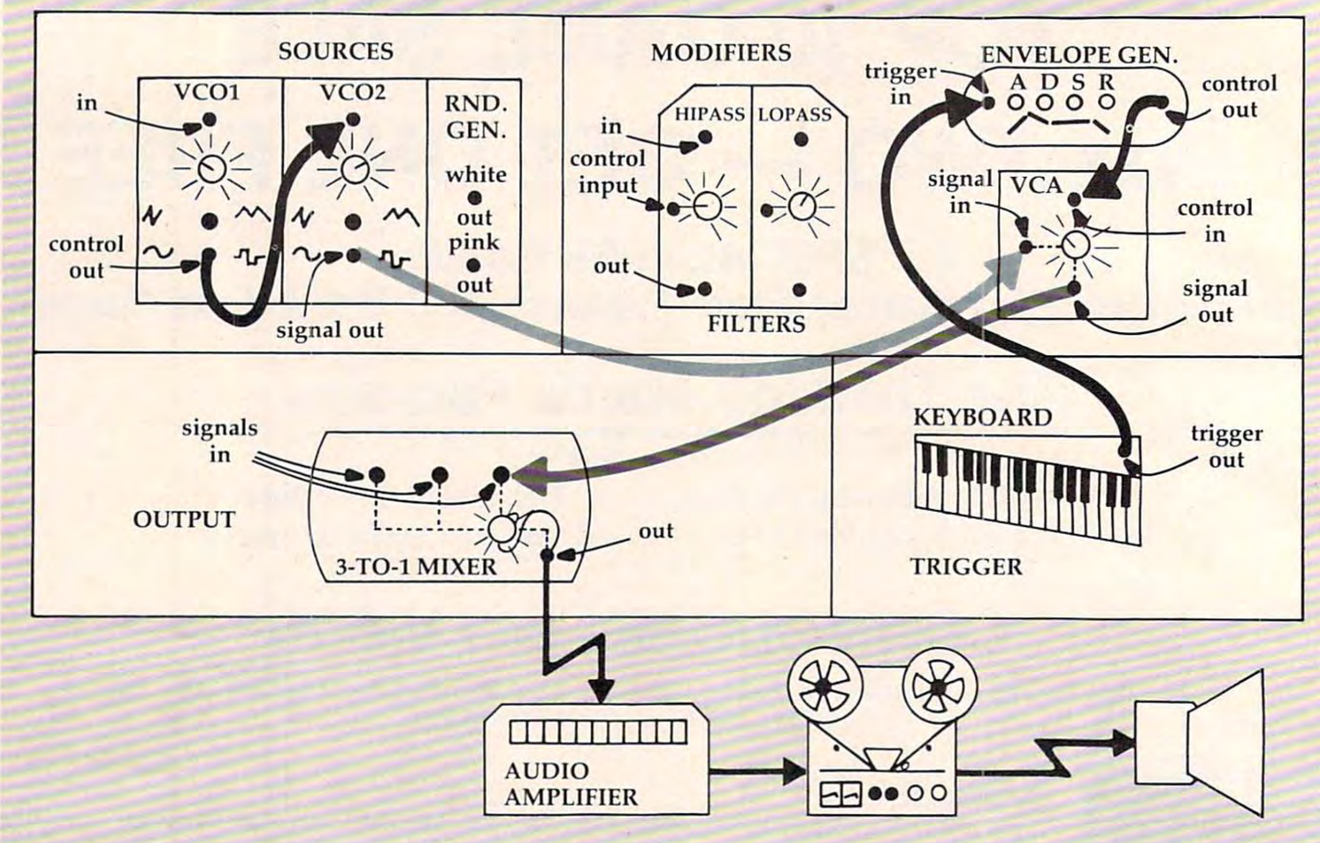


**JUMPMAN JUNIOR**
The devilish ALIENATORS are back! And they have overrun the Jupiter Command Substation. In this cartridge format sequel to the best-selling Jumpman, players must leap through 12 all new screens featuring electrocution traps, moving walls, hellstones, and dangers much too bizarre to be believed. How many screens can you master? Twelve different screens, 8 speeds.
List $39.95 **Sale $27.95**

***We have all Epyx games***

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

## Figure 5: Frequency Modulation




another provides incredibly powerful and varied tools for sound manipulation. Program 4 is one simple example of the FM technique.

## Program 4: Siren

```
300 FORI=ØTO24:POKE54272+I,Ø:NEXT
310 POKE54276,33:REM SAWTOOTH WAVE OSC1

320 POKE54286,3:REM CONTROL FREQ. OSC3

330 POKE54290,16:REM TRIANGLE WAVE OSC3

340 POKE54296,175:REM FULL VOL.& SELECT B
    AND-PASS & DISC. OSC3 FROM AUDIO

350 POKE54295,1:REM NO RES'NCE & CHOOSE O
    SC1 FOR FILTER
360 POKE54293,255:POKE54294,78:REM CUTOFF
     FREQUENCY
370 POKE54278,240:REM FULL SUSTAIN/FASTES
    T RELEASE RATE
375 FORT=1TO300
380 F=20000+PEEK(54299)*20:REM ADD OSC3 O
    UTPUT TO BASE FREQUENCY
390 HF=INT(F/256):LF=F-256*HF:REM SPLIT N
    EW FREQUENCY INTO HIGH/LOW BYTES

400 POKE54272,LF:POKE54273,HF:REM SET NEW
     OSC1 FREQUENCY
410 NEXT:POKE54276,32:POKE54296,Ø
420 END
```

The third oscillator on the SID chip is our control oscillator, as VCO1 is in Figure 5. We get access to a value corresponding to the wave shape of Oscillator 3 in register 27 (54299). If Oscillator 3 is set to a triangle wave, the values in register 27 will go up from 0 to 255 and then down from 255 to 0 in a symmetrical rhythm.

This is a nice shape for a siren sound, which is what Program 4 creates. Notice that the frequency of Oscillator 3 in line 520 is very low. This value allows the tracing of the waveform to be heard as a siren. The range of frequencies under approximately 32Hz is called the sub-audio range and refers to the fact that the actual waveform at these frequencies is discernible as individual pulses instead of as a continuous tone. When Oscillator 3's frequency is increased into the audio range (above about 29), the quality of the resulting tone becomes enjoyably less predictable.

Try POKEing 220 into 54286 at line 320 and running the routine. Note how the information in register 27 (54299) is utilized in line 380. It is increased by a factor of 20 and then added to the base frequency of 20000. Program 4 also uses a Band Pass filter, but for no particular reason other than simply to stick one in. Try a different value for the waveform in line 330. If you use 64 as your value, be sure to add a line to set Oscillator 3's pulse width.

The techniques of sound manipulation described above as used with an analog synthesizer have perhaps given you a better picture of the

# 64 WE DON'T CLAIM ARCADE WE ARE ARCADE

## KOOKY CLIMBER

As Alpine Joe, world famous mountain climber you must climb to the top of your apartment building. But watch out for Mr. Boofoo and his trained attack duck, Walter.

An incredible mixture of sound and graphics makes this an all time favorite.

Hires sprites, extended real time music and Arcade quality graphics.

*100% MACHINE CODE* DISK **29.95**


## POGO

- *100% MACHINE CODE*
- *ARCADE QUALITY GRAPHICS*
- *EXTENDED REAL TIME MUSIC*
- *EXCITING ANIMATED SPRITES*

DISK **36.95**


## SHOOTING GALLERY

- *100% MACHINE CODE*
- *REAL TIME MUSIC*
- *COLORFUL SCROLLING GRAPHICS*
- *GOOD PLAY ABILITY*

*CASSETTE* *DISK* **27.95**

## STAR SLAYER


Sent on a mission, you must destroy all the enemy base stations. But your task won't be easy, they have their best fighter pilots on your tail and they are hard to shake.

—And don't forget about asteroids — they are just as dangerous.

Never before has there been such realism and excitement in one game. STAR SLAYER brings you and your 64 into real-time space combat.

DISK **36.95**

*100% MACHINE LANGUAGE*

## WE ARE RENSOFT SOFTWARE SYSTEMS

*SPECIFY CASSETTE OR DISK*
*N.Y. RESIDENTS PLEASE ADD 7¼% SALES TAX*
*PLEASE ADD $2.00 SHIPPING AND HANDLING.*
*PREPAID (CHECK OR MONEY ORDER) OR C.O.D. ONLY*

**Contact Your Local Dealer.**
Or Send Check or Money Order For the Total

DISTRIBUTORS AND DEALER INQUIRES INVITED

*SOFTWARE AUTHORS WANTED*
DEALER INQUIRIES INVITED

P.O.Box 170 Coram
New York 11727
*(516) 732-3390*

---

# SCRIPT 64™

## THE WORDPROCESSOR THAT TURNS YOUR COMMODORE 64™ INTO A BUSINESS MACHINE!

- Flexible . . . Now, 80-columns without any additional hardware!
- Powerful . . . Supports all Commodore disk drives and IEEE printers thru optional C64-LINK interface.
- Versatile . . . Provides full-function text editing plus a built-in dictionary!

All this for only $99.95!

**Contact Your Nearest Commodore Dealer Today . . .**
**You'll Be So Glad You Did!**

**Distributed By:**
**COMPUTER MARKETING SERVICES INC.**

300 W. Marlton Pike
Cherry Hill, New Jersey 08002
(609) 795-9480

Commodore 64 is a trademark of Commodore Electronics Limited    Script 64 is a trademark of Richvale Telecommunications workings of the SID chip. As you learn more about the internal registers which control other functions, you will discover others just as interesting as those discussed here.

Get a copy of the *Commodore 64 Programmer's Reference Guide* and read about Ring Modulation, Filtering, and other advanced techniques. Sound effects are the most directly useful sound patches to work with at the start. Program 5 is an example of one I used for a Hangman program: it's the sound of nails being driven into wood. Imagine the other sound effects you can create for new game ideas.

The *User's Guide* and *Programmer's Reference Guide* have suggested patches for you to try out. Put some FOR..NEXT loops in, as we did in line 260 of Program 3, to have the computer, in effect, "adjust the pots" for you, as it alters individual registers. Once you've found a patch you like, save the register values for future reference. As you become more acquainted with the ways that sounds can be altered, you will find yourself noticing the subtler shades of sound color. You'll also begin to really know how the sounds on a TV commercial, videogame, or science fiction movie are created.

### Program 5: Driving Nails Into Wood

```
700 FORI=ØTO24:POKE54272+I,Ø:NEXT
710 CT=Ø
720 POKE54278,5:REM SUSTAIN/RELEASE
730 POKE54277,5:REM ATTACK/DECAY
740 POKE54276,129:REM NOISE WAVEFORM
750 POKE54295,241:REM RES'NCE & VOICE
760 POKE54293,54:POKE54294,28:REM CUTOFF
770 READA:REM INPUT HI BYTE FREQ. VALUE
780 READB:REM INPUT LO BYTE FREQ. VALUE
790 IFB=-1THEN9ØØ:REM BRANCH ON END-OF-DA
    TA
8ØØ POKE54273,A:POKE54272,B:REM SET FREQ
81Ø FORT=1TO35:POKE54296,79:NEXT:REM TURN
     ON VOLUME & FILTER
82Ø POKE54276,128:REM RELEASE CYCLE
83Ø GOTO73Ø:REM GET NEW NOTE
84Ø DATA17,37,19,63,21,154,22,227,25,177,
    28,214,32,94,34,175,34,255
845 DATA-1,-1
9ØØ CT=CT+1:IFCT+1<6THENRESTORE:FORT=1TO1
    ØØ*CT:NEXT:GOTO77Ø
91Ø POKE54296,Ø:REM TURN OFF VOLUME
```

©

# Atari Safe RAM

E. H. Foerster

*Are you tired of waiting while your BASIC program initializes? Would you like to save two-thirds of the memory required to store the DATA statements for your USR routines, character sets, or player/missile data? Would you like to save data generated by your program along with the program using just SAVE commands? This article will show you how to create protected and SAVEable RAM within the BASIC token program. Several USR routines are also included to allow you to add or delete safe RAM, move data in memory, and point a string to any place in memory. Also, the game "SKI!" illustrates these techniques in a practical, executable program. And each of these utilities does its job in a fraction of a second.*

The Atari 400/800 computers provide page six as free RAM to be used by BASIC. However, to use this RAM, the information must be stored in DATA statements in the BASIC program and then transferred to RAM using READ and POKE loops. This often delays the start of a program while the initialization routine is processed. You are also limited to 256 bytes when using page six.

A technique commonly used to circumvent this problem is to put the desired file in a string at the beginning of the string array storage area (STARP). The pointer to STARP is then moved past the desired file. When the program is SAVEd, the file is SAVEd along with the program. This technique requires special programming not only to save the file, but also to move the pointer back after the program is LOADed into the computer. You must insure that the pointers are not moved a second time when a program is executed following an initial run.

But there is another location within the BASIC token program that can be used to save and store data. The procedure we'll discuss will provide all the RAM you need at this location and protect it from any additions, deletions, or changes in the program. This protection, of course, does not extend to direct POKEs to this area.

No special routine will be needed to protect this RAM area once it has been inserted in the program. The RAM area and the information stored in it will remain there even when the program is SAVEd. The information is available immediately after the program is loaded and can be moved rapidly from this location to anywhere in memory using the MOVE DATA routine provided in this article.

## Using The Pointers

Before explaining the technique, let's briefly review the BASIC token program. There are several sections to every BASIC program, each with a two-byte pointer to a particular memory address. The address of the location is obtained by multiplying the second byte by 256 and adding the first byte. The location, name, and purpose of the pointers for the Atari BASIC token program are:

**128,129 LOMEM:** A 256-byte section used by BASIC for temporary storage.

**130,131 VNTP:** A table containing a list of all the variable names.

**132,133 VNTD:** Ending address of variable name table, plus one, one byte containing a zero when fewer than 128 variables have been used.

**134,135 VVTP:** Variable value table containing values for scalar variables and offset dimension, and length for arrays and strings.

**136,137 STMTAB:** Statement table containing the tokenized version of program statements.

**138,139 STMCUR:** Current statement being executed.

**140,141 STARP:** String array storage area.

When a program is SAVEd, LOMEM is subtracted from each of the pointers, and these values are SAVEd. Along with these pointers, all the information from VNTP up to STARP is SAVEd. When a program is LOADed, the reverse process occurs. This process is necessary because the addition of peripheral devices such as disk drives changes the value of LOMEM and thus the location of the BASIC program.

These pointers and the contents of the various sections are not static, but are moved around in memory as additions and deletions are made. The use of a new variable name – even in direct mode (no line number) – creates changes in the program even though no changes in the listed program occur. The new name is added to the variable name table. The pointers from VNTD to end are incremented by the length of the name and the contents of VNTD to end-of-program are moved up in memory by the length of the name. Then eight bytes are reserved at the end of the variable value table, and the contents and pointers for STMTAB to end-of-program are similarly moved.

## RAM Utilities

If we want to create RAM space for our own use within the BASIC program, several requirements must be satisfied. The area must be located in a place where it will be SAVEd when the program is SAVEd. The area must be in a location where any changes in the program (either in the direct mode or program mode) will not only leave the contents of the area intact, but also move the contents when the adjoining area is moved. Since the RAM area is not static, we also must have a pointer to this area. An area created between VNTD and VVTP is the only area that meets these requirements. Because VNTD is always one byte long, the location of safe RAM can be calculated from the formula:

**PEEK(132)+1+256*PEEK(133)**

Enter and run Program 1, and the routine for adding RAM along with four other utilities will be installed in safe RAM at location VNTD+1. You can now delete all program lines and save these utilities in a blank program for future use. You may want to leave a REM statement just to remind you what is in the program. The utilities and the calls for these utilities are as follows:

**Add RAM:** A=USR(VNTD+1, bytes of RAM)

**Delete RAM:** A=USR(VNTD+12, bytes of RAM)

**Move Data:** A=USR(VNTD+23,destination, source,bytes)

**String Assign:** A=USR(VNTD+90,ADR (string),dimension,length,desired location) where VNTD=PEEK(132)+256xPEEK(133)

You need to be careful when using these and other machine language routines. A mistake in any of the USR statements will more than likely lock up your computer. So before executing one of these calls, double-check your typing. If it is part of a program, SAVE the program before running it. Then if the computer locks up, shut it off, LOAD your program back in, and correct the mistake. No test is included in these routines to check for the correct number of arguments in the USR statement. If only one or some of the routines are needed for your program, then only those portions can be retained. We'll talk more about this later.

If you look at Program 1, you will notice that the Add RAM and Delete RAM routines are each only 11 bytes long. How do we go about moving all those pointers and memory with just 11 bytes of code? Why not let BASIC do it for us? The memory manager routines are located in the Atari BASIC cartridge at location 43135-43358 ($A87F-$A95E). All our USR routine does is load the 6502 registers with the appropriate values and then jump to the memory manager routines.

## Routine Features

The Add RAM routine cannot create RAM and is therefore limited by the amount of memory in your computer. If you attempt to add more bytes than are available, an "ERROR 2 AT LINE X" message is returned. Memory is always added at the current location of VVTP. Thus, any additional RAM is added at the end of current safe RAM.

The Delete RAM routine similarly deletes RAM at the end of current safe RAM. However, if you try to delete more bytes than were previously added, you might as well start all over.

The Move Data routine lets you move a block of data rapidly from one place to another in memory, including safe RAM. For example, moving 1K of data takes about 13 seconds using a PEEK and POKE routine. Using the Move Data routine, this same move takes a fraction of a second.

The move routine can even move a section of memory into an adjacent overlapping area without erasing any part of it. To demonstrate this point, execute the following line:

**DIM A$(9):A$="ABCDEF":A$(4)=A$:?A$**

The result: A$="ABCABCABC". What happens is that the origination string is changed by the destination string during the transfer of data. However, the Move Data routine moves bytes starting with either the first or last byte of the section of code to be moved, depending on whether the movement is forward or backward in memory. If you execute the following line:

```
DIM A$(9):A$"ABCDEF":A=USR(PEEK(132)+23
  +256xPEEK(133),ADR(A$)+3,ADR(A$),6):?A$
```

the resulting A$ equals "ABCABCDEF". So A$(4) actually is equal to the original A$.

It is possible to use this routine for many purposes including player/missile movement. I have not tested to see what would happen if the routine were called from a Vertical Blank Interrupt. Since the routine uses part of the memory manager routine and some of BASIC's zero page, it might interfere with the BASIC program. If this occurs, then the Vertical Blank Interrupt must first save locations $99-9C and then restore them at the end of the routine.

## Safe Strings

Safe RAM may also be used for storing strings using string manipulation techniques. The String Assign routine readily performs that task for us. Usually strings are manipulated by changing values in the variable value table. Bytes two and three must be changed to contain the low and high bytes of the offset of the location of the address of the string from STARP. Safe RAM, however, is located before STARP, and the offset from STARP is negative. To get a positive offset for a string located before STARP, we must add 65536 to the location before subtracting STARP and converting to a low and high byte offset. We are simply assuming that memory wraps around to zero when top-of-memory is reached.

Before we get too involved in how to calculate offset and where to POKE the values, let's look at the String Assign routine. This routine will calculate the offset for you and POKE it, the current length, and dimension values into the corresponding bytes in the variable value table for the variable identified by ADR(string). All you need to do is provide the address where the routine is located, the dimension, the length, and where you want the string to be located.

Before going to the String Assign routine, you must DIM the string to avoid getting an ERROR 9 message. But to save memory in the string array area, DIMension the string to length one. Then call the routine and change to the desired dimension. The term ADR(string) in the USR call is not used directly, but is used by the routine to identify the string to be changed. If you have ever tried string manipulation by calculating and POKEing string offsets, then you will certainly appreciate this routine.

String Assign is best used in a program. If used in direct mode, it is best done with all statements in a single, direct mode line. If, for example, you locate A$ at location 1536, the address for A$ will remain at 1536 while the direct line is being executed or while a program is running. But as soon as the program returns with a READY, the address for A$ will change.

There is a good reason for this. Any line that is typed in direct mode is entered as the last line in the statement table. STARP is moved backwards or forwards in memory, depending on the length of the line. Along with STARP, all pointers (offsets) for strings and arrays are changed accordingly. A string residing in the normal string array area can still be printed. However, a string moved outside that area will have only its offset changed, but not its location.

This routine can be used to clear memory for player/missile graphics. Point string A$ to the location of the player/missile area and set length and dimension to the number of bytes to be cleared with the String Assign routine. Then follow with the statement:

```
A$(1,1)=CHR$(0):A$(2)=A$
```

## Deleting The Utilities

If, after finishing a program, you want to delete part or all of the utility routines, you must perform several steps:

**1.** Move the utilities to location 1536+1 (page six). Then use the USR routines in page six to perform the remaining steps. Substitute 1536 for VNTD in the USR calls.

**2.** Move the routines you want to keep to the beginning of safe RAM.

**3.** Delete the remaining safe RAM.

**4.** Change all lines in the BASIC program that contain references to safe RAM to the new location within safe RAM (e.g., the first argument of the USR routines).

**5.** SAVE the program.

Safe RAM can be used for USR routines, player/missile data, custom character sets, and strings. But certain rules must be followed. For example, only relocatable USR routines can be left in safe RAM. Others must be moved to their designated location. If you want to change a program containing a USR routine, try the program by leaving it in safe RAM. If it does not work there, transfer it to the prescribed location using the Move Data routine. Character sets must be located on a 1K or 1/2K memory boundary and must be moved from safe RAM to their correct location.

Even though data must be moved from safe RAM to another location, you will still save more memory than when using DATA statements. For example, the 133 bytes of data in Program 1 occupy a total of 555 bytes when stored in the DATA statements. This is because each digit of each number and each comma occupies one byte of memory in the BASIC token program.

## A Practical Application For Safe RAM

As an example of the improvements which can be made using this technique, I have converted

"SKI!" (Atari version of "Slalom") from February COMPUTE! to initialize from safe RAM (see Program 2). The 15-second initialization has been cut to just a fraction of a second. And preloading the data into safe RAM saves 930 bytes more than the original initialization routine.

Other additions include a USR routine to generate the course, with the option of viewing the course before running it. Control of the skier's horizontal motion has been added to the Vertical Blank routine. This makes horizontal motion proportional to the scrolling rate. These changes do not affect the nearly instantaneous start of the program.

To enter the modified version, first type in the BASIC loader for SKI!. The DATA statements contain the safe RAM utilities and initialization data for the game. Each DATA line includes a checksum, which should greatly reduce the chance of errors. When you RUN the loader, it will POKE the data into safe RAM, then erase most of itself. Delete the remaining lines and, for safety, SAVE a copy of the safe RAM portion. Then type in modified SKI!. You must SAVE, not LIST, the combined program to tape or disk, since a LISTed version will not include the safe RAM portion. If you typed in the original SKI!, you'll be amazed at the increase in speed when you RUN the new version.

Using safe RAM and the utilities given in this article, you should be able to write programs that do not start with the message "Just a Moment" or "Initializing." The uses for safe RAM are not limited, of course, to the examples we've discussed. There's a lot of room for you to develop your own applications for safe RAM.

## Explanation Of USR Routines

**Add RAM (11 bytes):**
The routine uses a JMP $A881 to the memory manager routine for moving pointers and contents of token program to a higher location in memory. Before jumping, the following registers are loaded: X = token file pointer, which in our case is $86(VVTP); A = MSB(length); Y = LSB(length). Jumping to $A871 automatically loads A = 0.

**Delete RAM (11 bytes):**
This routine is identical to the above, except for JMP $A8FD.

**Move DATA (67 bytes):**
The routine stores destination in $9B,9C and source in $99,9A. It then determines if the move is positive or negative. For positive direction, a routine at $A8E3 is used which requires that X = MSB (length) + 1, Y = LSB (length), $9A = $9A + MSB (length), and $9C = $9C + MSB (length). For a negative move, the routine at $A94C is used which requires X = MSB (length) + 1, Y = complement of LSB (length) and ($99),Y and ($9B),Y point to first byte of source and destination, respectively.

**String Assign (44 bytes):**
This routine first obtains the variable number for the desired string from the statement table and loads it into the accumulator. A JSR $AC28 returns with the address of the desired string in the variable value table in $9D,9E. The dimension and length are then pulled off the stack and stored in ($9D),7 through ($9D),4. The desired location is then pulled off, and the offset calculated and stored in ($9D),2 and ($9D),3.

### Program 1: Safe RAM

```
10 FOR A=1 TO 133:READ B:POKE 1536+A
   ,B:C=C+B:NEXT A:IF C<>18631 THEN
   PRINT "CHECK DATA STATEMENTS":END

20 REM ADD RAM
30 DATA 104,104,170,104,168,138,162,
   134,76,129,168
40 REM DELETE RAM
50 DATA 104,104,170,104,168,138,162,
   134,76,253,168
60 REM MOVE DATA
70 DATA 104,162,3,104,149,153,202,16
   ,250,56,165,155,229,153,165,156,2
   29,154,104,170,144,16,24,101,154,
   133,154,138
80 DATA 101,156,133,156,232,104,168,
   76,227,168,232,104,168,101,153,13
   3,153,176,2,198,154,152,24,101,15
   5,133,155
90 DATA 176,2,198,156,152,73,255,168
   ,200,76,76,169
100 REM STRING ASSIGN
110 DATA 104,104,104,160,4,200,177,1
    38,201,60,208,249,200,200,200,17
    7,138,32,40,172,160,7,104,145,15
    7,136,192,2
120 DATA 208,248,56,170,104,229,140,
    145,157,200,138,229,141,145,157,
    96
130 A=USR(1536+1,133)
140 A=USR(1536+23,PEEK(132)+1+256*PE
    EK(133),1537,133)
```

### Program 2: Safe RAM Application

**BASIC Loader For SKI!**

```
0 REM LOADER FOR 'SKI'
10 ? "JUST A MOMENT":DIM A$(746):A=1
   :B=0:C=20:FOR D=0 TO 36:GOSUB 70:
   NEXT D:C=6:GOSUB 70
20 IF B<>73882 THEN ? "CHECK ALL DAT
   A LINES":END
30 VNTD=PEEK(132)+256*PEEK(133)
40 A=USR(ADR(A$),746)
50 A=USR(ADR(A$)+22,VNTD+1,ADR(A$),7
   46)
60 GOTO 1000
70 E=0:FOR F=1 TO C:READ G:E=E+G:B=B
   +G:A$(A,A)=CHR$(G):A=A+1:NEXT F
80 READ F:IF F<>E THEN ? "CHECK DATA
    STATEMENTS AT LINE ";100+D*10:EN
   D
90 RETURN
```

# "For Your Eyes Only"

Consider the possibility that your personal data is available to anyone with some computer knowledge. Protect your software system with

**Softloc** T.M. ©1983

**A software protection system for the Commodore 64.**

CommodoreTM 1541 disk drive and C2N tape unit required

**3 types of protection in one package**

- 7 digit access code
- encoded programs
- modified diskette directory

**Suggested list price, $29.95**

Softloc™ is available on disk at local Commodore™ dealers.

## Softpeople, Inc.

631 Cleveland Avenue So.
Saint Paul, MN 55116
1-800-447-3273
Dealer Inquiries Invited

*Commodore 64 is a trademark of Commodore Business Machines, Ltd. Softloc is a trademark of Softpeople, Inc.*

```
100 DATA 104,104,170,104,168,138,162
,134,76,129,168,104,104,170,104,
168,138,162,134,76,2617
110 DATA 253,168,104,162,3,104,149,1
53,202,16,250,56,165,155,229,153
,165,156,229,154,3026
120 DATA 104,170,144,16,24,101,154,1
33,154,138,101,156,133,156,232,1
04,168,76,227,168,2659
130 DATA 232,104,168,101,153,133,153
,176,2,198,154,152,24,101,155,13
3,155,176,2,198,2670
140 DATA 156,152,73,255,168,200,76,7
6,169,104,104,104,160,4,200,177,
138,201,60,208,2785
150 DATA 249,200,200,200,177,138,32,
40,172,160,7,104,145,157,136,192
,2,208,248,56,2823
160 DATA 170,104,229,140,145,157,200
,138,229,141,145,157,96,112,112,
112,70,155,34,102,2748
170 DATA 20,144,38,38,38,38,38,38,38
,38,38,38,38,38,38,38,38,38,38,3
8,848
180 DATA 38,38,6,65,130,9,0,0,0,21,0
,0,0,0,0,0,0,0,0,0,307
190 DATA 0,0,6,14,28,24,32,0,128,0,0
,0,0,0,0,0,0,0,0,0,232
200 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0
210 DATA 0,192,192,220,20,28,7,5,7,0
,0,24,52,44,60,24,0,16,56,56,100
3
220 DATA 124,124,254,16,16,8,28,62,6
2,62,8,8,0,0,56,94,106,94,116,56
,1294
230 DATA 0,0,119,69,117,21,119,0,0,8
,24,56,120,8,8,8,8,0,0,0,685
240 DATA 48,88,56,16,186,254,89,24,1
56,82,33,16,8,0,0,0,12,26,28,8,1
130
250 DATA 93,127,154,24,57,74,132,8,1
6,0,0,24,60,60,24,24,60,186,89,2
4,1236
260 DATA 154,170,198,65,65,1,18,36,7
4,161,18,156,77,10,24,24,0,0,0,0
,1251
270 DATA 0,169,0,133,0,169,1,141,99,
6,169,8,141,98,6,104,104,133,7,1
04,1592
280 DATA 133,6,104,104,133,1,162,10,
160,117,169,7,32,92,228,96,216,1
73,4,208,2155
290 DATA 240,4,169,0,133,0,165,0,240
,85,165,1,240,81,206,99,6,173,99
,6,2112
300 DATA 208,73,198,203,208,26,169,2
,133,203,173,124,2,208,3,206,100
,6,173,125,2543
310 DATA 2,208,3,238,100,6,173,100,6
,141,0,208,165,0,141,99,6,206,98
,6,1906
320 DATA 174,98,6,142,5,212,208,27,1
60,0,56,177,6,233,20,145,6,160,1
,177,2013
330 DATA 6,233,0,145,6,169,7,141,98,
6,141,5,212,198,1,76,98,228,0,0,
1770
340 DATA 0,0,104,162,228,160,98,169,
7,32,92,228,96,173,10,210,41,3,2
01,3,2017
350 DATA 176,247,96,170,169,72,224,1
,240,6,169,73,144,2,169,0,145,20
3,96,165,2567
360 DATA 206,133,205,173,10,210,41,7
,24,105,6,168,169,134,145,203,17
3,10,210,48,2380
370 DATA 14,169,23,133,207,169,18,13
3,208,169,22,133,209,208,12,169,
17,133,207,169,2522
380 DATA 22,133,208,169,18,133,209,5
6,152,229,209,168,162,3,152,24,1
01,207,168,169,2692
390 DATA 204,145,203,152,24,101,208,
168,169,204,145,203,202,208,235,
96,104,104,133,204,3212
400 DATA 104,133,203,169,0,133,205,1
69,24,133,206,32,113,6,168,32,11
3,6,32,123,2104
410 DATA 6,136,16,247,160,17,32,113,
6,201,1,240,4,200,144,1,200,32,1
13,6,1875
420 DATA 32,123,6,200,192,20,208,245
,173,10,210,201,13,176,18,24,165
,205,105,10,2336
430 DATA 197,206,176,9,32,139,6,240,
52,208,2,208,194,173,10,210,201,
25,176,4,2468
440 DATA 169,7,208,20,173,10,210,201
,25,176,4,169,10,208,9,173,10,21
0,201,25,2218
450 DATA 176,19,169,139,170,173,10,2
10,41,15,201,12,176,247,24,105,3
,168,138,145,2341
460 DATA 203,230,206,165,206,201,175
,208,1,96,24,169,20,101,203,133,
203,165,204,105,3018
470 DATA 0,133,204,208,182,0,727
1000 A=0:FOR B=1 TO 5
1010 GRAPHICS 0:POSITION 2,4
1020 FOR C=1 TO 10:? A*10:A=A+1:NEXT
     C
1030 ? "CONT":POSITION 2,0:POKE 842,
     13:STOP
1040 POKE 842,12
1050 NEXT B
1060 ?" SAFE RAM IS LOADED. DELETE
     REMAINING LINES AND ENTER MODI
     FIED SKI!"
```

## Modified SKI!

```
100 DIM SCREEN$(1),PM$(1)
101 DIM LEFT$(20),CENTER$(20),RIGHT$
    (20),CURR$(20),CRASH$(20),ERASE$
    (20),DIR(8),SCR(4),DLIST$(1)
102 DIM T$(20),TOPLINE$(20):GOTO 130
110 REM * SKI * LINE 100 MUST BE TYP
    EDIN FIRST!!!
120 HI=INT(A/256):LO=A-HI*256:RETURN
125 POKE 66,1:FOR W=1 TO 10:POKE 532
    79,0:POKE 53279,8:NEXT W:POKE 66
    ,0:RETURN
130 GOSUB 790:REM Initialization rou
    tines
140 REM PLAYER ROUTINE
150 POKE 559,62:POKE 54279,PMBASE
160 POKE 53277,3:POKE 704,2*16+6
170 PO=1024:YP=180:XP=128
180 PM$(PO)=CHR$(0):PM$(PO+254)=CHR$
    (0):PM$(PO+1)=PM$(PO)
195 SCR(0)=0:SCR(1)=10:SCR(2)=4:SCR(
    3)=2:SCR(4)=1
200 ERASE$=CHR$(0):ERASE$(20)=CHR$(0
    ):ERASE$(2)=ERASE$
```

```
210 LEFT$=ERASE$:CENTER$=ERASE$:RIGH
    T$=ERASE$:CRASH$=ERASE$
220 FOR I=Ø TO 15
230 LEFT$(I+2,I+2)=CHR$(PEEK(CHSET+2
    Ø8+I))
240 CENTER$(I+2,I+2)=CHR$(PEEK(CHSET
    +224+I))
250 RIGHT$(I+2,I+2)=CHR$(PEEK(CHSET+
    1Ø4+I))
260 CRASH$(I+2,I+2)=CHR$(PEEK(CHSET+
    24Ø+I))
270 NEXT I
280 DIR(Ø)=Ø:DIR(1)=2Ø:DIR(2)=19:DIR
    (3)=21:DIR(4)=1:FOR I=Ø TO 3:DIR
    (I+5)=-DIR(I):NEXT I:DIR(5)=-1
290 CURR$=CENTER$
295 POKE 1636,XP:POKE 2Ø3,2:SCR=Ø
300 PM$(PO+YP,PO+YP+2Ø)=CURR$
310 SCR=SCR+SCR(PEEK(Ø)):POKE 77,Ø
320 POSITION 2,Ø:? #6;SCR;" ";:POSIT
    ION 15,Ø:IF PEEK(Ø)<>Ø THEN ? #6
    ;(5-PEEK(Ø))*1ØØ
330 IF PEEK(1)<3 THEN POKE Ø,Ø:GOTO
    74Ø
340 ST=STICK(Ø)
350 LEFT= NOT PTRIG(1):RIGHT= NOT PT
    RIG(Ø):LR=LEFT+2*RIGHT
360 CURR$=CENTER$:XP=PEEK(1636)
370 IF LEFT THEN CURR$=LEFT$:IF LR<>
    OLR THEN SV=2:TI=5
380 IF RIGHT THEN CURR$=RIGHT$:IF LR
    <>OLR THEN SV=4:TI=5
390 IF TI>O THEN TI=TI-1:SOUND Ø,SV,
    Ø,TI
400 IF LR=Ø THEN SOUND Ø,Ø,Ø,Ø:TI=Ø
410 OLR=L
420 UP=(ST=14 OR ST=1Ø OR ST=6):DOWN
    =(ST=5 OR ST=9 OR ST=13)
430 YP=YP-2*UP+2*DOWN:IF YP>2ØØ THEN
     YP=2ØØ
440 IF YP<4Ø THEN YP=4Ø
450 POKE Ø,1+(YP>13Ø)+(YP>16Ø)+(YP>1
    85)
460 IF PEEK(PØPF)=Ø THEN 3ØØ
470 WHICH=INT(LOG(PEEK(PØPF))/LOG(2)
    +Ø.1):TEMP=PEEK(Ø):POKE Ø,Ø
480 PM$(PO+YP,PO+YP+2Ø)=ERASE$
490 POKE HITCLR,1:IF WHICH<>2 THEN 6
    2Ø
500 REM POINTS
510 PTR=ASC(DLIST$(8))+256*ASC(DLIST
    $(9))
520 LINE=INT((YP-39)/8)+1
530 COL=INT((XP-49)/8)+1
540 LOC=PTR+LINE*2Ø+COL:SOUND Ø,Ø,Ø,
    Ø
550 FOR I=Ø TO 8:P=PEEK(LOC+DIR(I))
560 IF P<128 OR P>192 THEN 59Ø
570 POKE LOC+DIR(I),O
580 SCR=SCR+(P=139)*5Ø+(P=134)*1ØØ*(
    5-TEMP):I=11:NEXT I:GOTO 6ØØ
590 NEXT I:GOTO 61Ø
600 FOR W=15 TO Ø STEP -1:SOUND Ø,2Ø
    ,1Ø,W:NEXT W
610 POKE Ø,TEMP:POKE HITCLR,1:GOTO 3
    ØØ
620 REM CRASH!
630 SOUND Ø,Ø,Ø,Ø
640 PM$(PO+YP,PO+YP+2Ø)=CRASH$
650 FOR W=1ØØ TO 15Ø STEP 2:SOUND Ø,
    W,12,1Ø:NEXT W
660 PM$(PO+YP,PO+YP+2Ø)=ERASE$
670 YP=2ØØ
680 PM$(PO+YP,PO+YP+2Ø)=CURR$
690 POKE O,1:SOUND Ø,Ø,Ø,Ø
700 XP=INT(72+9Ø*RND(Ø)):POKE 53248,
    XP:POKE 1636,XP
710 IF PEEK(PØPF)<>Ø THEN POKE HITCL
    R,1:GOTO 7ØØ
720 POKE HITCLR,Ø:SCR=SCR-5Ø:IF SCR<
    Ø THEN SCR=Ø
730 GOTO 3ØØ
740 IF SCR>HSCR THEN HSCR=SCR
745 POSITION 8,Ø:? #6;"  HIGH ";HSCR
750 SOUND Ø,Ø,Ø,Ø
760 SCREEN$(326,336)="press{,}start"
770 IF STRIG(Ø) THEN 77Ø
780 GOTO 13Ø
790 REM INITIALIZATION
800 GRAPHICS 17:HILO=12Ø:POKE 53248,
    Ø:POKE Ø,Ø
810 SETCOLOR 4,Ø,12:SETCOLOR 1,12,8:
    SETCOLOR 2,9,6:SETCOLOR Ø,15,4
820 PØPF=53252:HITCLR=53278:POKE HIT
    CLR,Ø
830 SCRBASE=PEEK(1Ø6)-16:REM 4K BOUN
    DARY
840 PMBASE=SCRBASE-8:REM 2K BOUNDARY
    ,DOUBLE-LINE RES
850 CHBASE=PMBASE:REM FILL UP OFFSET
     WITH CHARACTERS
870 VNTD=PEEK(132)+256*PEEK(133)
880 A=USR(VNTD+9Ø,ADR(SCREEN$),4Ø97,
    4Ø97,SCRBASE*256)
890 A=USR(VNTD+9Ø,ADR(PM$),2Ø49,2Ø49
    ,PMBASE*256)
900 A=USR(VNTD+9Ø,ADR(DLIST$),4Ø,4Ø,
    VNTD+134)
910 CHSET=CHBASE*256
920 A=USR(VNTD+23,CHSET,VNTD+174,12Ø
    ):A=USR(VNTD+23,CHSET+128,57472,
    344):A=USR(VNTD+23,CHSET+2Ø8,VNT
    D+294,48)
930 A=VNTD+377:GOSUB 12Ø:POKE VNTD+3
    68,HI:POKE VNTD+37Ø,LO
940 A=USR(VNTD+23,1649,VNTD+494,1Ø3)
950 Z=USR(VNTD+483):REM DISABLE VBLA
    NK
960 POKE 756,CHBASE:RESTORE 99Ø
980 A=ADR(DLIST$):GOSUB HILO:POKE 56
    1,HI:POKE 56Ø,LO
1020 DLIST$(32)=CHR$(PEEK(56Ø)):DLIS
     T$(33)=CHR$(PEEK(561))
1030 SCREEN$(1)=CHR$(Ø):SCREEN$(4Ø95
     )=CHR$(Ø):SCREEN$(2)=SCREEN$
1040 TOPLINE$=SCREEN$
1050 A=ADR(TOPLINE$):GOSUB HILO
1060 DLIST$(5,5)=CHR$(LO):DLIST$(6,6
     )=CHR$(HI)
1070 POKE 88,LO:POKE 89,HI
1080 POSITION 8,Ø:? #6;"SKI!";
1082 SCREEN$(121,139)="press{,}butto
     n{,}to{,}ski"
1083 SCREEN$(163,178)="pull{,}joysti
     ck{,}to":SCREEN$(185,195)="view
     {,}course"
1085 SCREEN$(4Ø3,419)="G{2 ,}I{2 ,}N
     {2 ,}I{2 ,}S{2 ,}R"
1090 A=USR(VNTD+597,ADR(SCREEN$)+48Ø
     )
1410 A=SCRBASE*256
1420 A=A-2Ø*(STICK(Ø)=14)+2Ø*(STICK(
     Ø)=13)
1430 IF A>SCRBASE*256+348Ø THEN A=A-
     2Ø
1440 IF A<SCRBASE*256 THEN A=A+2Ø
```

```
1450 GOSUB HILO:T$=CHR$(LO):T$(2)=CH
     R$(HI):DLIST$(8,9)=T$
1460 IF STRIG(Ø)=1 THEN 1420
1470 A=SCRBASE*256+3480:GOSUB HILO
1480 T$=CHR$(LO):T$(2)=CHR$(HI):DLIS
     T$(8,9)=T$
1481 GOSUB 125:IF STRIG(Ø) THEN 1481
1490 A=USR(VNTD+342,ADR(DLIST$(8)),1
     76)
1495 SCREEN$(121,195)=SCREEN$(12Ø)
1500 RETURN
2000 VNTD=PEEK(132)+256*PEEK(133)
2010 POKE VNTD+342+26,78
2020 POKE VNTD+342+28,67
2030 POKE VNTD+342+39,85
2040 POKE VNTD+342+43,81
2050 POKE VNTD+342+51,73
2060 A=USR(VNTD+23,2ØØØØ,VNTD+342,36
     5)
2070 A=USR(VNTD+23,2ØØØØ+52+3Ø,2ØØØØ
     +52,365)
2075 A=USR(VNTD+23,2ØØØØ+36+9,2ØØØØ+
     36,4ØØ)
2080 RESTORE 2ØØØ
2085 FOR A=Ø TO 8:READ B:POKE 2ØØ36+
     A,B:NEXT A
2086 DATA 173,4,2Ø8,24Ø,4,169,Ø,133,
     Ø
2090 FOR A=Ø TO 29:READ B:POKE 2ØØØØ
     +61+A,B:NEXT A
2095 DATA 198,2Ø3,2Ø8,26,169,2,133,2
     Ø3
2100 DATA 173,124,2,2Ø8,3,2Ø6,1ØØ,6
2110 DATA 173,125,2,2Ø8,3,238,1ØØ,6
2112 DATA 173,1ØØ,6,141,Ø,2Ø8
2200 FOR A=Ø TO 1Ø3:? A,PEEK(VNTD+36
     Ø+A),PEEK(2ØØØØ+A):NEXT A
```

©