New for the New Year

#6279 Introduction to Computer Animation
(Wadsworth) Now you can produce amazing computer graphics — even if you can't draw a straight line! Learn to draw lines and shapes, make graphs, draw pictures, and do animations with popular microcomputers such as the Apple II, the TRS-80, and the PET. Takes a step-by-step approach to using low-resolution graphics. Many program listings illustrate graphic techniques using a minimum of mathematics. **$9.95**

#6251 Create Word Puzzles with Your Microcomputer
(Mau) Create your own letter inserts, acrostics, cryptograms, word-finds, quote-falls, fill-ins, and other word puzzles. Contains BASIC programs for producing both blank puzzles or print-outs, following magazine format. Provides complete information for establishing and maintaining word and quotation files, techniques for producing complex puzzles, and serves as a tutorial on managing large text data bases. **$14.95**

#6256 Software Toolkit for Microcomputers: Improving Productivity with High-Level Languages and Operating Systems
(ed. Schindler) An edited compilation from *Electronic Design* magazine reflecting the importance of software in improving productivity. Stresses the importance of choosing the right language and discusses the advantages and disadvantages of high-level languages, with special attention to FORTRAN, COBOL, Forth, PL/M, Pascal, McPascal, and ADA — the operating language recently adopted by the Defense Department. **$14.95**

#5203 The Investor's Computer Handbook
(Packer) Manage your investments better using your micro! NO PREVIOUS MICROCOMPUTER EXPERIENCE IS NEEDED. Offers advice on choosing the system best suited for your particular investment applications, including information on hardware, software, peripherals, and a list of required and recommended components. **$11.95**

#5626 Basic Apple™ BASIC
(Coan) A complete guide to Applesoft BASIC. Takes you from beginning concepts, such as entering data and obtaining output, and planning programs, to more advanced topics, such as numeric and string arrays, and sequential and random access files. Alternate techniques for programming in Apple Integer BASIC are also covered, as well as low-resolution and high-resolution graphics. **$12.95**

#5204 CP/M Revealed
(Dennon) Intended for CP/M users interested in improving their skills, this is a guide to the CP/M operating system: the console monitor (CCP), the system manager (BDOS), and the input output driver package (CBIOS). Provides a clear understanding of the data structure of the CP/M disk and other essentials for using CP/M effectively. Covers buying CP/M, booting up, logging in, changing memory size, mapping disk space, calling all programs, and more. **$13.95**

#6278 Graphics Cookbook for the Apple
(Wadsworth) Use your Apple II to "paint" shapes, objects, and letters in low-resolution graphics. Contains complete annotated Applesoft BASIC programs to draw all pictures described in the book as well as suggestions for improving programming techniques. **$9.95**

Apple is a registered trademark of Apple Computer Company, Inc.; CP/M is a registered trademark of Digital Research Corp.; none is affiliated with Hayden Book Company, Inc.

Hayden

Order Today

Available at your local computer store or order by phone 1-800-631-0856
operator CO 23
In New Jersey
(201) 843-0550 ext. 382

VISA MasterCard

Mail to:

Dept. # CO 23
Hayden Book Company, Inc.
50 Essex Street
Rochelle Park, NJ 07662

Method of payment:
☐ My check or money order is enclosed.
Please charge to my:
☐ Visa ☐ MasterCard
We pay postage and handling.

Residents of NJ and CA add sales tax.

Please send me the item(s) indicated below by code number.

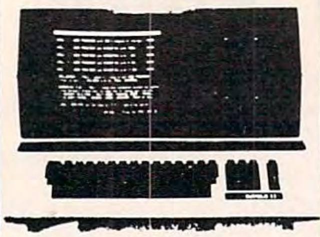
☐ ☐ ☐

If I am not completely satisfied, I may return the book(s) undamaged within 10 days for a complete refund.

Mr./Ms. ____
Address ____ Apt. # ____
City ____ State ____ Zip ____
Visa/MasterCard # ____ Exp. Date ____
Signature ____ Prices subject to change

409

EAGLE



64K Ram
780 KB Disk Storage
Word Processing, Ultracalc CP/M
C-Basic Software
Smith Corona TP 1
Letter Quality Printer

$2995.00

Retail Value $4895.00

EAGLE 1600... CALL

TELEVIDEO TERMINALS

910	$579.00
912C	$699.00
920C	$749.00
925C	$749.00
950	$950.00

TELEVIDEO COMPUTERS

800A	$1319.00
802	$2649.00
802H	$4695.00
806	$5495.00
816	$9495.00
803	CALL
1603	CALL

MONITORS

AMDEK

100 B & W	$74.95
300G	$169.00
300A	$179.00
Color I	$339.00
Color II	$699.00
Color II A	$799.00
Color III	$399.00
Color IV	CALL

BMC

12" Green	$79.99
13" Color 1401 (Mid Res.)	$369.00
9191U 13"	$329.00

ZENITH

ZVM 121	$99.00

SHARP

Sharp 13" Color TV	$275.00

PANASONIC

TR-120MIP (High Res. Green)	$159.00
CT-160 Dual Mode Color	$299.00

HEWLETT PACKARD

hp 41CV

$209

HP 41C	$149.00
HP 10C	$69.00
HP 11C	$79.00
HP 12C	$114.00
HP 15C	$109.00
NEW 16C	$114.00

PERIPHERALS

HP41 Card Reader	$144.00
HPIL Module	$99.00
HPIL Cassette	$449.00
HPIL Printer	$419.00
Quad Memory Module	$64.00
Time Module	$64.00
Extended Function Module	$64.00

NEC

COMPUTERS

8001A	$729.00
8031	$729.00
8012	$549.00

PRINTERS

8023	$499.00
7710/7730	$2399.00
3510/3530	$1599.00

MONITORS

JB-1260	$129.00
JB-1201	$159.00
JC-1201	$319.00
JC-1203	$729.00

TIMEX SINCLAIR 1000

$89.99

16K Memory Module	$44.95
Vu-Calc	$17.95
Super Math	$12.95
Check Book Manager	$13.95
The Organizer	$14.95
The Budgeter	$13.95
Stock Option	$14.95
Loan & Mortgage Amortizer	$12.95

SHARP

PC-1500 POCKET COMPUTER $209

CE 150 Printer, Plotter and Cass. Interface Unit... $172.00
CE 152 Cass. Recorder... $69.00
CE 155 8K Ram Expansion Module..... $94.00

HP●85 $1969

HP 125	$1999.00
HP 85 16K Memory Module	$169.00
5¼" Dual Floppy Disk	$1799.00
Hard Disk w/Floppy	$4349.00
Hard Disk	$3549.00
"Sweet Lips" Printer	$1219.00
80 Column Printer	$649.00

IBM®

NEC

3550 PRINTER...$2099

PERCOM DRIVES

5¼" 160K Disk Drive	$249.00
5¼" 320K Disk Drive	$299.00

AMDEK

310A Amber Monitor	$179.00
310G	$179.00
Amdisk (3¼" Drive)	$729.00
DXY Plotter	$759.00
Color II	$699.00

SOFTWARE

I.U.S. Easywriter II	$249.00
I.U.S. Easyspeller	$129.00
Peach Package (GL/AP/AR)	$419.00

PROFESSIONAL SOFTWARE

IBM/PC Word Processing	$319.00

PRINTERS

SMITH CORONA

TP 1	$599.00

C. ITOH (TEC)

Starwriter (F10-40CPS)	$1399.00
Printmaster (F10-55CPS)	$1749.00
Prowriter 80 Col. (Parallel)	$499.00
Prowriter 80 Col. (Serial)	$629.00
Prowriter 2 (132 Col.)	$799.00

OKIDATA

82A	$429.00
83A	$659.00
84 (Parallel)	$1079.00
84 (Serial)	$1199.00

IDS

MicroPrism	$649.00
132 (Fully Configured)	$1599.00
80 (Fully Configured)	$1399.00

Call for other configurations.

STAR

Gemini 10	$379.00

DAISYWRITER

Letter Quality	1049.00

DIABLO

620	$1179.00
630	$1849.00

MODEMS

HAYES

Smart	$239.00
Smart 1200 (1200 Baud)	$549.00
Chronograph	$199.00
Micromodem II (with Term)	$309.00
Micromodem 100	$309.00

NOVATION

Cat	$144.00
D-Cat	$159.00
212 Auto Cat	$589.00
Apple Cat II	$279.00
212 Apple Cat II	$609.00

CALL for Price and Availability on New **NOVATION** Cat 103, 103/212 and J-Cat.

ANCHOR

Mark I (RS-232)	$79.00
Mark II (Atari)	79.00
Mark III (TI-99)	109.00
Mark IV (CBM/PET)	$125.00
Mark V (OSBORNE)	$95.00
Mark VI (IBM-PC)	$179.00
Mark VII (Auto Answer Call)	$119.00
TRS -80 Color Computer	$99.00
9 Volt Power Supply	$9.00

commodore

8032	$1039.00
CBM 64	CALL
4032	$749.00
8096 Upgrade Kit	$369.00
Super Pet	$1499.00
2031	$469.00
8250 Dbl. Sided Disk Drive	$1699.00
D9060 5 Meg. Hard Disk	$2399.00
D9060 7.5 Meg. Hard Disk	$2699.00
8050	$1299.00
4040	$969.00
8300 (Letter Quality)	$1549.00
8023	$599.00
4022	$399.00
New Z-Ram, Adds CP/M & 64K	$549.00
The Manager	$209.00
Magis	CALL
Word Pro 5 Plus	$319.00
Word Pro 4 Plus	$299.00
Word Pro 3 Plus	$199.00
The Administrator	$379.00
Info Pro Plus	$219.00
Power	$79.00
CBM 8032 Dust Cover	$14.99
CBM 8050/4040 Dust Cover	$10.99

computer mail order east

800-233-8950

IN PA. CALL (717)327-9575, 477 E.THIRD ST., WILLIAMSPORT, PA. 17701

In stock items shipped same day you call. No risk, no deposit on C.O.D. orders. Pre-paid orders receive free shipping within the continental United States with no waiting period for certified checks or money orders. Add 3% (minimum $3.00) shipping and handling on all C.O.D. and Credit Card orders. NV and PA residents add sales tax. All items subject to availability and price change. **NOTE:** We stock manufacturer's and third party software for most all computers on the market! CALL TODAY FOR OUR NEW CATALOGUE.

FRANKLIN

ACE 1000
ACE 10 with Controller Card
ACE Writer Word Processor
CALL...
FOR SYSTEM PRICE!
ACE 1200........CALL

PERCOM

DISK DRIVES FOR ATARI

AT 88-S1	$399.00
AT 88-A1	$289.00
RFD 40-S1	$539.00
RFD 40-A1	$329.00
RFD 40-S2	$869.00
RFD 44-S1	$659.00
RFD 44-S2	$999.00

RANA DISK DRIVES

Call for price and availability on the new Rana Disk Drives for The Apple and Franklin Computer Systems.

ATARI HOMECOMPUTERS

400

16K.........$199
32K........$274*
48K........$359*
*Non-Atari Ram

800

48K.....$499
New low price effective January 1, 1983.

Call for Price and Availability of the NEW
64K ATARI 1200

410 Recorder	$74.00
810 Disk Drive	$429.00
822 Printer	$269.00
825 Printer	$589.00
830 Modem	$159.00
820 Printer	$259.00
850 Interface	$169.00
CX40 Joy Sticks (pair)	$18.00
CX853 Atari 16K Ram	$77.95

Axlon Ramdisk (128K)	$429.95
Intec 48K Board	$159.00
Intec 32K Board	$74.00
One Year Extended Warranty	$70.00
CX481 Entertainer Package	$69.00
CX482 Educator Package	$130.00
CX483 Programmer Package	$54.00
CX484 Communicator Package	$344.00

VISICORP

for Apple, IBM & Franklin

Visidex	$189.00
Visifile	$189.00
Visiplot	$159.00
Visiterm	$89.00
Visitrend/Plot	$229.00
VisiSchedule	$229.00
Desktop Plan	$189.00
Visicalc (Apple II, Atari, CBM, IBM)	$179.00

Visicorp prices for IBM may vary slightly.

CONTINENTAL

Home Accnt. (Apple/Franklin)	$59.00
Home Accountant (IBM)	$119.00
1st Class Mail (Apple/Franklin)	$59.00

SIRIUS

Free Fall	$24.00
Beer Run	$24.00
Snake Byte	$24.00
Space Eggs	$24.00
Sneakers	$24.00
Bandits	$28.00

BRODERBUND

Apple Panic	$23.00
David's Magic	$27.00
Star Blazer	$25.00
Arcade Machine	$34.00
Choplifter	$27.00
Serpentine	$27.00

INFOCOM

Deadline	$35.00
Star Cross	$29.00
Zork I	$29.00
Zork II or III	$29.00

MPC

Bubdisk (128K Ram)	$719.00

AXLON

Ram Disk (Apple/Franklin)	CALL

Call for Price on
VIC 64

Peripherals and Software.

PROFESSIONAL SOFTWARE
Word Processing for VIC 64....$79.95

μ-SCI

MICRO-SCI DISK DRIVES FOR APPLE & FRANKLIN

A2	$299.00
A40	$349.00
A70	$459.00
C2 Controller	$79.00
C47 Controller	$89.00

FLOPPY DISKS

MAXELL

MD I (Box of 10)	$32.00
MD II (Box of 10)	$44.00
FD I (8")	$40.00
FD II (8" DD)	$50.00

VERBATUM

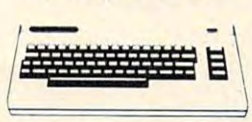
5¼" SS DD	$26.00
5¼" DS DD	$36.00

ELEPHANT

5¼" SS DD	$19.99

VIC 20

$179.



VIC 20 Dust Cover	$9.99
VIC 1530 Datassette	$69.00
VIC 1540 Disk Drive	$339.00
VIC 1541 (64K Disk Drive)	CALL
VIC 1525 Graphic Printer	$339.00
VIC 1210 3K Mem. Exp.	$32.00
VIC 1110 8K Mem. Exp.	$53.00
VIC 1111 16K Mem. Exp.	$94.00
VIC 1011 RS232C Term. Interface	$43.00
VIC 1112 IEEE-488 Interface	$86.00
VIC 1211 Super Expander	$53.00
VIC Mother Board	$99.00

SOFTWARE FOR ATARI

ATARI

Pac-Man	$33.00
Centipede	$33.00
Caverns of Mars	$32.00
Asteroids	$29.00
Missile Command	$29.00
Star Raiders	$35.00
Galaxian	$33.00
Defender	$33.00

ON-LINE

Jawbreaker	$27.00
Softporn	$27.00
Wizard and the Princess	$29.00
The Next Step	$34.00
Mission Asteroid	$22.00
Mouskattack	$31.00
Frogger	$31.00
Cross Fire (ROM)	$36.00

SYNAPSE

File Manager 800 +	$69.00
Chicken	$26.00
Dodge Racer	$26.00
Synassembler	$30.00
Page 6	$19.00
Shamus	$26.00
Protector	$26.00
Nautilus	$26.00
Slime	$26.00
Disk Manager	$24.00

DATASOFT

Pacific Coast Highway	$25.00
Canyon Climber	$25.00
Tumble Bugs	$25.00
Shooting Arcade	$25.00
Clowns and Balloons	$25.00
Graphic Master	$30.00
Graphic Generator	$13.00
Micro Painter	$25.00
Text Wizard	$79.00
Spell Wizard	$64.00
Bishop's Square	$25.00
Sands of Egypt	$25.00

APX

Text Formatter	$18.50
Family Budgeter	$18.50
Eastern Front	$24.00
Family Cash	$18.50
Jukebox	$13.00
Downhill	$18.50
Outlaw	$18.50
Holy Grail	$24.00
Player Piano	$18.50
Keyboard Piano	$18.50
Number Blast	$13.00
Frogmaster	$18.50
747 Land Simulator	$18.50
Word Processor	$40.00

EPYX

Crush, Crumble & Chomp	$24.00
Crypt of the Undead	$24.00
Curse of Ra	$16.00
Datestones & Ryn	$16.00
Invasion Orion	$19.00
King Arthur's Heir	$24.00
Morloc's Tower	$16.00
Rescue at Rigel	$24.00
Ricochet	$16.00
Star Warrior	$29.00
Temple of Asphai	$29.00
Upper Reaches of Apshai	$16.00

CBS

K-razy Shoot Out	$32.00
K-razy Kritters	$32.00
K-razy Antics	$32.00
K-star Patrol	$32.00

STICK STAND $6.99

Arcade Action from your ATARI or VIC Joy Stick

computer mail order west

800-648-3311

IN NV. CALL (702)588-5654, P.O. BOX 6689, STATELINE, NV. 89449

INTERNATIONAL ORDERS: All shipments outside continental United States must be pre-paid by certified check only! Include 3%(minimum $3.00) shipping and handling.
EDUCATIONAL DISCOUNTS: Additional discounts are available from both Computer Mail Order locations to qualified Educational Institutions.

maze is constructed. The actual time needed to construct the mazes varies greatly; a one-level maze takes approximately 30 seconds, while a 32-level maze will take approximately 16 minutes to build. For a rough estimate of the time you have for a coffee break, divide the number of levels entered by two. The result is the approximate time needed by the computer, in minutes.

Once the computer has completed construction of the maze, the screen is turned back on, and you are asked to PRESS START TO BEGIN. Watch the word START closely. See how it is flashing on and off? This effect is produced by toggling CHACT in rapid succession (alternately POKEing in one and two). You are asked to press START when you are ready because the program times you, and it would not be fair to start timing from the moment the maze was completed.

Therefore, when you are ready to begin, you press START, which tells the program that you are poised with joystick in hand; the top level is displayed and timing begins. You will see an "S" in the upper left corner of the screen, with the ball character (control-T) underneath. *You* are the ball character.

Threading The Tunnels

Just move the joystick in the direction you want to go. "Sure," you say, "but where *do* I want to go?" Simple enough. If you chose a one-level maze (chicken!), you will see an "F" at the lower right corner of the maze. That's where you want to go. If you were gutsy, however, and chose any number of levels greater than one, you will see five graphics "+" characters at random points throughout the top level. These symbols represent tunnels, through which you must pass to reach the finish (which is always in the lower right of the bottom level of the maze). As you might have guessed, you always start at the upper left of the top level.

To pass through a tunnel, simply move onto the "+" symbol and press the "fire" button. Violà! The new level is displayed instantly. Have you gone up or down? Well, if you were on the top level, the only place you could go is down. If you are in the middle of a maze of four or more levels, then I have absolutely no idea which direction you'll go; you may pass through the same level three or four times before you realize that you've gone nowhere.

In mazes of ten or more levels, be prepared to see the same level a few times before you make any progress. No matter how many levels you chose, however, the goal is still the same. You must try to go down to unexplored levels; if you end up on a level you have been on already, you have looped, and you must figure out whether you've gone up or down.

In any case, find the "F" on the lowest level, go to the space directly above it, and move down. If you do not push the joystick down, the timer will continue, and your record time will be lost. When the timer has stopped, you will hear five beeps.

If you do not hear the five beeps, you have not stopped the timer or the sound is gone on your machine. Either way, just remember to go down when you reach the finish – as you get better and better, times will get tougher and tougher to beat, and each second will become important.

That's all there is to it. After the five beeps have informed you that the timer has stopped, the screen will become visible (no change for visible mazers), and the time used to complete the maze will be displayed in hours:minutes:seconds format. The program will loop until you press the START button again, which will cause the program to re-RUN.

Possible Dead Ends

There are a few caveats, however. First, if you are attempting an invisible maze, some joystick directions may not work. There is nothing wrong with the program; if you cannot move in a certain direction, you have hit a wall (I told you they were still there!). Second, don't even try to do deep invisible mazes without the consent of your psychologist. Third, each tunnel can be used only once, so make your moves wisely.

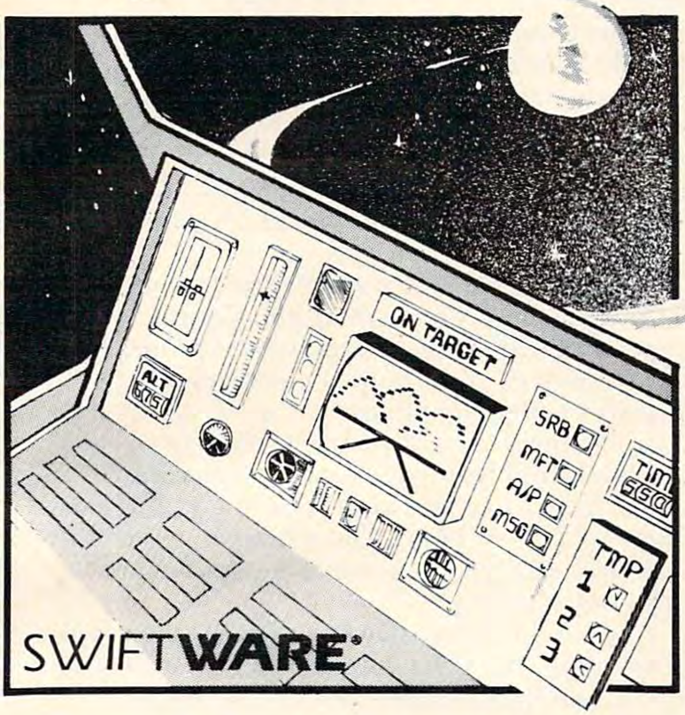
Last, and most important, don't *ever* remove lines 14 and 15. This program, as mentioned earlier, will cause the computer to do some strange things if you hit the BREAK key. Lines 14 and 15 turn off the BREAK key; the only way to get out of the program is to hit the SYSTEM RESET button.

The Program

Now let's look at how the program accomplishes what it does. Line 8 is self-explanatory. Line 10 resets the screen and sets the variable TOP to the address of the LSB of the screen memory address. By POKEing different numbers into TOP and TOP+1, we can display any area of memory. Line 12 stores the value of SAVMSC (locations 88 and 89, 58 and 59 hex) into RL and RH, respectively. This step is necessary to reset the destination of PRINT statements after these locations have been modified by the maze generator routine.

Lines 20 and 23 obtain the required data from the user and determine the value of BOT, the page number of the lowest memory address to be used. Line 25 makes sure that we haven't used up all available memory, and informs the user of any memory conflict. Line 27 lets the user know that the delay which will follow is intentional, not

Fly the SPACE SHUTTLE from your ATARI®



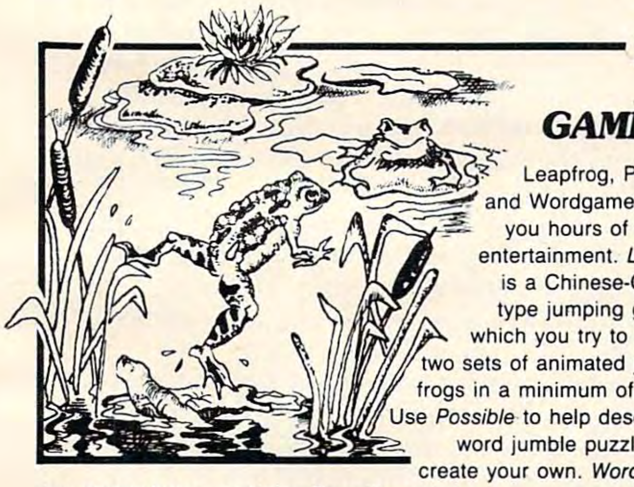
SPACE SHUTTLE

By Paul Kindl

Join the crew of the *Space Shuttle* as they prepare to take the next step into the world of space travel. Take control of the world's first reuseable spaceship, the *Space Shuttle,* and in an accurate full graphic simulation, place yourself in the cockpit. Pilot the Space Shuttle through take-off with booster stage separations, orbit, descent down the glide path and landing to touchdown – complete with a chase plane and scrolling runway visible through the cockpit windscreen. You assume command throughout all phases of the mission aided by complete instrumentation.

Price: $29.95 32K Diskette

ATARI® is a registered trademark of Atari Inc., a Warner Communications, Co.



FUN 'N' GAMES™

Leapfrog, Possible, and Wordgames giving you hours of fun and entertainment. *Leapfrog* is a Chinese-Checker type jumping game in which you try to position two sets of animated jumping frogs in a minimum of moves. Use *Possible* to help descramble word jumble puzzles or to create your own. *Wordgames,* two games in one, contains *Guessit* – a deductive alphabetic reasoning game for one or two players and *Wordjumble* - a multiple word descrambling puzzle with play-on-word hints and mystery answers. Disk version of *Guessit* works with a Votrax Type'N'Talk. A real crowd pleaser. Joystick and printer optional.

By Jerry White

Price: $17.95 16K Cassette/ $19.95 24K Diskette

HAUNTED HILL ™

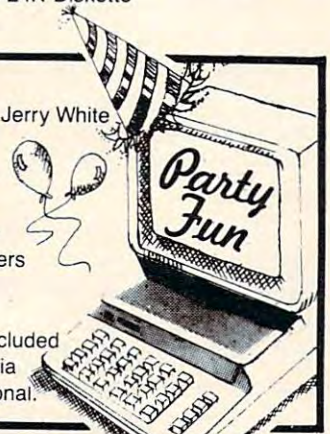
Fight bats and ghosts in the dark of the cemetary. This exciting, all machine language game has arcade quality graphics and speed. Requires Joystick.

By George Richardson

Price: $24.95 16K Cassette/ $29.95 16K Diskette

TRIVIA TREK™

By Jerry White



Play it for fun, test your knowledge or entertain friends. Fun for the whole family and great at parties. A multiple choice trivia game for one or two players complete with 500 trivia questions in fifty categories and two thousand multiple choice answers. Included is a program for creating your own trivia questions and answers. Joystick optional.

Price: $29.95 32K Diskette

Plus many more

AVAILABLE AT SELECT COMPUTER STORES

MAIL ORDERS: Send check or money order plus $2.50 shipping and handling. N.Y. Residents add 7¼% sales tax.

TELEPHONE ORDERS: (516)549-9141

Dealer Inquires Invited

Send for FREE catalog

MasterCard

SWIFTY SOFTWARE, INC.
64 Broad Hollow Road
Melville, New York 11747

VISA

something gone wrong with the program.

Line 28 turns off the screen and sets up the display for the start of the game. Line 29 employs a trick described by Bill Wilkinson in "Insight: Atari" (**COMPUTE!**, May 1982, #24), for clearing memory using the CLEAR key. Line 31 establishes the top of maze memory and sets up a loop to construct each of the MAXLEV levels of the maze. Lines 40-111 constitute the maze generator routine by Charles Bond.

Establishing Start And Finish

Line 120 restores the PRINT statement destination to its original value by POKEing RL and RH back into SAVMSC. Line 130 establishes the "S" in the upper left and the "F" in the lower right of the maze. Line 135 checks to see if any tunnels have to be built; in other words, if the maze is only one level, jump over the tunnel building routine (lines 140-170).

The tail end of line 170 restores the screen and sets up the console switches for reading. Line 172 executes a GOSUB to the routine that randomly sets the color of the background at the beginning and also each time the user passes through a tunnel. Line 173 loops indefinitely until the user presses the START button. This line is the one that toggles CHACT, as described earlier.

Line 174 makes the maze visible or invisible, based on your response to the second prompt at the beginning of the program. Line 175 resets the three-byte timer RTCLOK to zero. Line 180 determines the start position for the player and tells the display list where the first level of the maze is. Lines 185-321 are the main loop and should be self-explanatory.

A few notes, though: line 190 reads the joystick and the trigger, lines 200-230 perform routine motion, line 235 checks for a win, line 240 checks for walls, and lines 300-321 change levels. Lines 400-415 stop the timer, sound the bell, and display the time used. Line 420 sets up the console switches for reading and POKEs a 124 into the attract mode flag ATRACT (location 77, hex 4D). The 124 in ATRACT gives the user approximately 16 seconds before the screen goes into attract mode.

Line 430 loops until the START button is pressed. Line 450 is the string A$ (we can't PRINT it because we've changed the screen memory locations). Don't forget to put the exclamation point towards the end of the line; doing that fools BASIC into reading trailing blanks to fill up A$. Finally, line 500 reads a random number from the random number generator RANDOM (location 53770, D20A hex), masks out the four low-order bits, and uses it to set the background color. If you're interested in the technical aspects of the game, read on. If not, RUN the program and have some fun.

Inner Secrets Of Page Flipping

The programming tool behind the entire program is called page flipping. What this technique involves is changing the address that the ANTIC chip reads to determine the start of screen memory. This address is always in the display list, which is pointed to by SDLSTL and SDLSTH (locations 560 and 561, hex 230 and 231) in standard LSB, MSB order.

In the display list you will find all sorts of numbers; all have a meaning and should not be tampered with by the inexperienced programmer. In different graphics modes, the display list changes both in length and location.

In general, the display list follows two rules. First, all graphics modes accessible through BASIC have display lists that start with 112, 112, 112 in three successive bytes. These three bytes tell the ANTIC that there are to be 24 blank lines on the television screen.

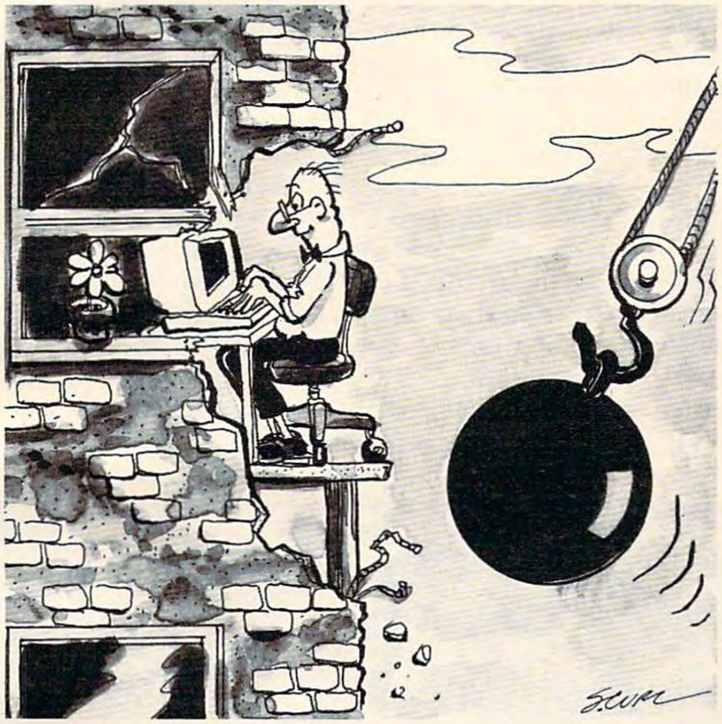
Second, the fourth location of the display list contains a byte which has its sixth bit set. The rest of the byte varies depending on the graphics mode, but bit six is always set. Bit six, when set, tells the ANTIC chip that it is to begin direct memory access (DMA) at the location pointed to by the next two bytes. Therefore, any area in memory can be displayed by POKEing the address (LSB, MSB) into the location pointed to by SDLSTL and SDLSTH plus four.

This is the basis of this program. All screens are constructed before play begins, and, instead of drawing an entire new screen, all the program does is change these addresses to point to the first byte of the new screen.

During the blank-out period at the start of the program, the entire maze is constructed, layer by layer, and the resulting mazes are stored in 1K decrements, starting with the last free kilobyte memory block before the display list. The maze generator routine does not even need to be modified for this purpose; all that was done was to change the PRINT destination pointer SAVMSC (location 88, hex 58, mentioned earlier). In other words, all I did was fool the maze generator routine into thinking that screen memory was located in middle area RAM (instead of the top), and since 960 bytes are needed for the standard GRAPHICS 0 screen, 1K blocks were very convenient.

The tunnels used this information both at construction time and at level-changing time. Random numbers were all that was necessary to build the tunnels; checks were required only to make sure that the tunnels would be within the maze and that they did not cut through maze walls. Since no other checks are made, it is possible

Q: What is the hardest thing in the world?
A: Tearing yourself away from an EPYX game.



Nobody but nobody builds more lasting playing value into their computer games than EPYX.

EPYX—computer games thinkers play. EPYX—producers of award-winning games (including "Crush, Crumble and Chomp"; "Dragon's Eye"; "Temple of Apshai"...and dozens of other thinking games.) For Atari,* Apple,* Commodore,* Radio Shack,* and IBM* personal computers. EPYX—the leader in computer game quality, creativity, innovation.

You will love them all.

Write or phone for our latest catalog; it's absolutely free...we even pay the postage. Or stop in at your favorite computer dealer. He should have a supply of catalogs and he can also show you some of those marvelous EPYX games in action.

*The trademarks, respectively, of Atari, Inc.; Apple Computer, Inc.; Commodore International; Tandy Corp.; International Business Machines. And EPYX is the trademark of Automated Simulations, Inc., just so you don't forget.

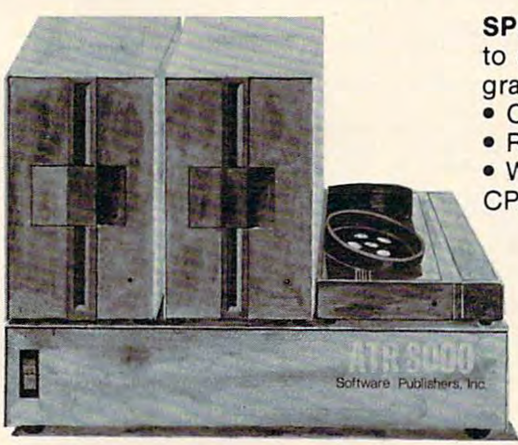
EPYX

AUTOMATED SIMULATIONS
1043 Kiel Court
Sunnyvale, California
(408) 745-0700

ATR8000: THE VERSATILE 4MHz Z80 DOUBLE DENSITY DISK INTERFACE FOR THE ATARI® 800/400

Connect the ATR8000 to convert your ATARI 800/400 into a viable business machine. Open the doors to a further dimension by adding the 64K CP/M upgrade—then you can enter the vast CP/M marketplace to select programs tailored to your specific needs.

The ATR8000 is versatile, so it can grow as you need it to. Its handling of disk drives is revolutionary . . . it runs four 5¼" or 8" drives, single or double or quad density, single or double-sided *AND* allows you to mix them!



SPECS: 4MHz Z80 processor • 16k RAM standard • Connects to expansion port of the ATARI 800/400 • 12½" x 11½" x 2½" gray and beige enclosure • Runs four drives of mixed definition • Centronics parallel and RS-232 serial port (and printer drivers) • Runs single density ATARI DOS and existing ATARI software • With OSA+, Ver. 4, is double density • 64K upgrade includes CP/M.

PRICING:

ATR8000	$499.95	OSA+, Ver. 4	$49.95
5¼" Drive	$399.95	Printer Cable	$29.00
64K Upgrade		5¼" Drive Cable	$35.00
(with CP/M)	$250.00	8" Drive Adapter	—CALL—

CONTACT: SOFTWARE PUBLISHERS, INC.
2500 E. Randol Mill Rd., Suite 125
Arlington, TX 76011
(817) 469-1181

ATARI is a registered trademark of ATARI, Inc. CP/M is a registered trademark of Digital Research, Inc.

to have many tunnels packed closely together.

The simple method of checking upward and downward movement causes tunnels to be disabled as they are used. When the player changes levels, a control-T character is left where the graphics plus symbol was previously. As a result, the checks for the graphics plus symbols will always fail on an already-used tunnel. This feature, added only to make the mazes more challenging, can easily be altered by changing the GOTO 185 in line 250 to GOTO 190.

This simple change makes the program think that you have just moved across or down (i.e., you have not changed levels). Therefore, the program replaces the previous space with the variable T, which contains the screen memory value of the space you were on before. When you move, the control-T is moved in the proper direction, and T is POKEd into the space you just moved from. It is confusing, but it works, and it works fast.

Tunnel Checking

The tunnels, when used, merely change the value of the sixth byte of the display list. Since 1K memory blocks are used, it is not necessary to change the fifth, LSB of the display list DMA address; it will always be zero. Either the sixth byte is added to four, or four is subtracted from it. The reason for this change should be evident – four pages constitute one kilobyte of memory.

Locating the mazes in this fashion greatly simplifies all checks. Instead of going through a series of different LSB, MSB checks to determine the location (two-dimensionally) of a space on two different levels, all that is required is a PEEK to the address plus 1024 (1K) and the address minus 1024. Again, this is how tunnel checking is done in lines 305 and 310.

Last, let's look at the timer. From the time the computer is powered up until the time it is powered down, the OS, as part of its stage one

Mazemaking For VIC, 64, PET/CBM, And Other Microsoft BASIC Computers

In the December 1981 **COMPUTE!**, we published one of the most useful (and deceptively short) subroutines for game-lovers of all ages. "Mastermaze" for the Atari is based upon Charles Bond's original idea that a random maze of any size could be created quickly right on the screen.

For those who might have missed this excellent subroutine, the basis for all kinds of games, the version in Program 1 below can be used by any computer with Microsoft BASIC where you can POKE to the screen memory. As listed, it will work on Commodore VIC, 64, and PET/CBM's. You need to know the number of columns on your screen and the memory address of the start of the screen RAM memory (the listing contains this information for Commodore computers).

This maze generator can get you started toward programming a variety of entertaining and challenging games. It always results in a maze with only one significant pathway to the solution and it will always fill the screen with pathways.

The short additional routine (Program 2) creates a semi-intelligent "mouse" that runs through the maze, attempting to solve it as best it can. Add it to the maze generator and, when the maze is drawn, hit any key. See if you can tell what rules the mouse uses to find the solution to the maze.

If you come up with an interesting game based on this generator, send it in to **COMPUTE!** and if we think others will enjoy it, we'll print it.

Program 1.

```
100 DIMA(3): REM SET UP DIRECTION TABLE
110 A(0)=2:A(1)=-80:A(2)=-2:A(3)=80:REM VALUES
     FOR 40 COLUMN SCREEN
111 REM FOR 80-COLUMN SCREEN CHANGE: A(1)=-160
    : A(3)=160
112 REM FOR THE VIC 22 COLUMN SCREEN CHANGE: A
    (1)=-44: A(3)=44
120 WL=160:HL=32:SC=32768:A=SC+81: REM CHARACT
    ER, SCREEN, & START
121 REM FOR UNEXPANDED VIC USE SC=7680:A=SC+45

130 PRINT"{CLEAR}": REM CLEAR SCREEN AND GENER
    ATE MAZE BACKGROUND FIELD
140 FORI=1TO23
150 PRINT"{REV}
              ":REM 22,40, OR 80 SPACES
151 REM PRINT THE CORRECT AMOUNT OF RVS SPACES
     TO MAKE A SCREEN LINE WHITE.
160 NEXT I
200 REM GENERATE MAZE
210 POKEA,4
220 J=INT(RND(1)*4):X=J
230 B=A+A(J): IF PEEK(B)=WL THEN POKE B,J:POKE
    A+A(J)/2,HL:A=B:GOTO220
240 J=(J+1)*-(J<3):IFJ<>XTHEN230
250 J=PEEK(A):POKEA,HL:IFJ<4THENA=A-A(J):GOTO2
    20
300 REM MAZE IS DONE.  WAIT FOR A KEY TO BE PR
    ESSED.
310 GETC$:IFC$=""THEN310
```

Program 2.

```
1000 REM MAZE MOUSE
1010 POKEA,81:J=2
1020 B=A+A(J)/2:IFPEEK(B)=HLTHENPOKEB,81:POKEA,
    HL:A=B:J=(J+2)+4*(J>1)
1030 J=(J-1)-4*(J=0):GOTO1020
```

VBLANK (vertical blank) routine, increments the three-byte jiffy counter RTCLOK. RTCLOK is located in three consecutive bytes starting at address 18 decimal, 12 hex.

Unlike most of the system numbers, this clock is stored in MSB first, LSB last order. Since vertical blanks occur once every sixtieth of a second, this timer counts "jiffies" (sixtieths of a second). When the game start is pressed, zeros are POKEd into the clock addresses (line 175). As soon as the player has completed the maze, the locations are read and stored in the variable ET (for elapsed time). Simple mathematical manipulations derive the hours, minutes, and seconds and store them in the variables EH, EM, and ES, respectively.

That's all there is to it. Since we know that we started at zero, no other manipulations are needed. (Incidentally, it is possible to stop the clock, but doing so requires a shutdown of the entire system VBLANK routine, which can have disastrous effects on your computer.)

And there's the entire program. If you have any questions or if you would like me to make a cassette copy of the program, send a cassette, a self-addressed, stamped mailer, and $3 to:

Ken Szajda
59 West Lakeshore Drive
Rockaway, NJ 07866

```
8 DIM A(3),A$(37):SW=0
10 GRAPHICS 0:TOP=PEEK(560)+256*PEEK(
   561)+4
12 RL=PEEK(88):RH=PEEK(89)
14 O=PEEK(16)-128:IF O<0 THEN O=O+128
15 POKE 16,O:POKE 53774,O
20 ? :? "# OF LEVELS";:INPUT MAXLEV:M
   AXLEV=MAXLEV-1:? "INVISIBLE (1) OR
    VISIBLE (2)";:IF MAXLEV<0 THEN MA
   XLEV=0
23 BOT=INT(TOP/256)-MAXLEV*4-4:INPUT
   INV
25 IF BOT*256<PEEK(144)+256*PEEK(145)
    THEN ? "***INSUFFICIENT MEMORY***
   ":GOTO 20
27 ? "{CLEAR}":POKE 755,1:POSITION 4,
   10:? "CONSTRUCTING MAZE...PLEASE W
   AIT":FOR DEL=0 TO 1000:NEXT DEL:PO
   KE 755,2
28 POKE 559,0:? "{CLEAR}":POSITION 10
   ,11:? "PRESS START TO BEGIN"
29 TM=PEEK(106):POKE 106,TM-6:POKE 88
   ,0:POKE 89,BOT:? "{CLEAR}":POKE 10
   6,TM
30 R1=BOT+MAXLEV*4:FOR X=BOT TO R1 ST
   EP 4:POKE 77,0:POKE 88,0:POKE 89,X
40 REM MAZE GENERATOR ROUTINE BY C. B
   OND
50 A(0)=2:A(1)=-80:A(2)=-2:A(3)=80:B=
   0
60 SC=PEEK(88)+256*PEEK(89):A=SC+43
65 POSITION 2,0:POKE 752,1:FOR I=1 TO
    23:? "{37 SPACES}":NEXT I
70 POKE A,5
80 J=INT(RND(0)*4):X1=J
90 B=A+A(J):IF PEEK(B)=128 THEN POKE
   B,J+1:POKE A+A(J)/2,0:A=B:GOTO 80
100 J=(J+1)*(J<3):IF J<>X1 THEN 90
110 J=PEEK(A):POKE A,0:IF J<5 THEN A=
    A-A(J-1):GOTO 80
111 IF J=128 THEN STOP
120 NEXT X:POKE 88,RL:POKE 89,RH
130 POKE BOT*256+917,38:POKE R1*256+3
    ,51
135 IF MAXLEV=0 THEN POKE 559,34:POKE
     53279,8:GOTO 172
140 FOR X=BOT TO R1-4 STEP 4:FOR Y=1
    TO 5
150 J=INT(RND(0)*876)+43
151 W=J-(INT(J/40)*40):IF W<3 OR W=39
     THEN 150
155 IF PEEK(X*256+J)=0 AND PEEK(X*256
    +1024+J)=0 THEN POKE X*256+J,83:P
    OKE X*256+1024+J,83:GOTO 170
160 GOTO 150
170 NEXT Y:NEXT X:POKE 559,34:POKE 53
    279,8
172 GOSUB 500
173 IF PEEK(53279)<>6 THEN POKE 755,-
    PEEK(755)+3:GOTO 173
174 POKE 755,INV
175 POKE 18,0:POKE 19,0:POKE 20,0
180 ST=R1*256+43:WIN=BOT*256+960:POKE
     TOP,0:POKE TOP+1,R1
185 S=PEEK(ST):T=ST:POKE ST,84
190 Q=STICK(0):R=STRIG(0):IF R=0 AND
    S=83 THEN 300
200 IF Q=7 THEN ST=ST+1
210 IF Q=11 THEN ST=ST-1
220 IF Q=14 THEN ST=ST-40
230 IF Q=13 THEN ST=ST+40
235 IF PEEK(ST)=38 THEN 400
240 IF PEEK(ST)=128 OR PEEK(ST)=51 TH
    EN ST=T
250 IF ST<>T THEN SW=0:POKE T,S:POKE
    77,0:GOTO 185
251 GOTO 190
300 IF SW=1 THEN 190
305 IF PEEK(ST+1024)=83 THEN R1=R1+4:
    ST=ST+1024:GOTO 320
310 IF PEEK(ST-1024)=83 THEN R1=R1-4:
    ST=ST-1024
320 IF R1<BOT OR R1>MAXLEV*4+BOT THEN
     330
321 POKE TOP+1,R1:SW=1:GOSUB 500:GOTO
     185
400 ET=PEEK(18)*65536+PEEK(19)*256+PE
    EK(20):EH=INT(ET/216000):EM=INT((
    ET-EH*216000)/3600)
401 FOR X=1 TO 5:FOR Y=15 TO 0 STEP -
    0.2:SOUND 0,9,10,Y:NEXT Y:NEXT X:
    POKE 755,2
402 ES=INT((ET-EH*216000-EM*3600)/60)
403 ? "{CLEAR}":? :? "445 DATA ELAPSE
    D TIME: ";EH;":";EM;":";ES;"
    {19 SPACES}|"
404 ? "CONT":POSITION 0,0:POKE 842,13
    :STOP
405 POKE 842,12
406 POSITION 2,15:RESTORE :FOR Y=0 TO 1
410 READ A$:FOR X=BOT*256+Y*40 TO BOT
    *256+Y*40+LEN(A$)-1:POKE X+2,ASC(
    A$(X-BOT*256+1-Y*40,X-BOT*256+1-Y
    *40))-32
415 NEXT X:NEXT Y
420 POKE 53279,8
430 IF PEEK(53279)<>6 THEN 430
440 RUN
450 DATA PRESS START FOR ANOTHER MAZE
    {10 SPACES}"
500 AA=PEEK(53770):AB=AA-(INT(AA/16)*
    16):SETCOLOR 2,AB,4:POKE 712,PEEK
    (710):RETURN
```

©

Making Change

Myron Miller

"Making Change" is an educational program to teach children the concept of using quarters, dimes, nickels, and pennies to make a given amount of change. The program uses 3K RAM memory and will work on the TRS-80 Color Computer, PET/CBM, Apple, Atari, and VIC computers.

This program first asks for the user's name and then presents the first problem. There are two types of problems which are alternately displayed. All odd-numbered problems begin like this:

1 JOHN GIVE ME 68 CENTS.
HOW MANY QUARTERS?

One is the problem number, John is the user's name, and 68 is a random integer between and including 1 and 100.

The player must enter how many quarters there would be in the requested amount. The program will then ask for dimes, nickels, and pennies in the same manner. For each type of coin the user must enter the number of coins and press RETURN. If a certain coin is not needed, the user should enter 0 and press RETURN. The total value of the user's answer should equal the requested amount (for 68 cents: 2 quarters, 1 dime, 1 nickel, and 3 pennies would be entered).

Even-numbered problems look like this:

2 JOHN I HAVE:
3 QUARTERS,
1 DIMES,
0 NICKELS, AND
4 PENNIES.
HOW MUCH CHANGE DO I HAVE?

The even problems present the opposite case. The user must add up the change and enter the total amount. Again, RETURN must be pressed after the entry. The total amount will always be in the range of 1 to 100 cents since both types of problems use the same program line to generate a random integer.

For both types of problems, the program checks the user's answer. If the answer is correct, the program will so indicate and will go on to the scoring routine. If the answer is wrong, the program will print out X CENTS SHORT JOHN, or X CENTS TOO MUCH JOHN. The youngster should be encouraged to use this information to correct the answer, for the problem will repeat up to three additional times. If the answer is still wrong, the program will display the correct answer and will move on to scoring.

Reward Or Penalty

The program keeps track of two independent scores: conventional and reward. The conventional score is similar to a test score used in schools. It records how many problems were done, how many were correct, how many were wrong, and gives a percentage of correct answers. The conventional score is applied only to the first presentation of the problem; that is, repeat problems are not counted in the conventional score.

The reward score tries to motivate the user by paying one cent for every correct answer. To keep things fair, it charges one cent for every wrong answer. Thus the user earns money for right answers, but loses money for wrong answers. The reward score is applied to the repeat problems as well as to the first presentation. The reason for this is to encourage the user to take the repeat problems seriously. There can be a difference between the two scores because the conventional is applied only to the first attempt.

In the odd problems, the program will reject an answer given in all pennies (38 pennies for 38 cents) for any amount greater than four cents. It will also reject a fractional answer (3.8 dimes for 38 cents). In either case, the user is fined one cent for cheating. This should take care of any get-rich-quick schemes. The even problems will not accept a decimal answer (.38 for 38 cents). The concern here is to avoid round off errors in floating point numbers, not cheating. Thus the score is not affected.

How To Encourage The Player

There are some changes and improvements that can be added. If your child is having a rough time with the program, I would recommend deleting line number 4100 from the program. This removes the "money lost" counter used by the reward score. The reward score can be brutal to a youngster having difficulty. Each problem has a potential earning of one cent, but a potential loss of four

cents. A child can lose a lot more money than he or she earns. We want to encourage the youngster, not chip away at self-esteem. Take out 4100 and the player can earn money, but not lose it (except for fines). Kids will learn far more if you let them win something.

If you enjoy programming graphics (I don't), you may want to liven up the program. Add graphics only for correct answers; don't make it interesting to get the problem wrong. For both odd and even problems, the program will go to line 3000 if the correct answer is given. Insert your graphics in lines 3001 to 3899; this space was left open for that purpose.

Kids will learn far more if you let them win something.

As the program uses no PEEKs, POKEs, or machine language, it is easy to modify. Original ROM PET machines need to have line number 540 changed to:

540 X% = 100*RND(-TI) + 1

No other changes should be necessary. On the VIC, some of the printed lines will exceed the screen's 22 columns. You will have to break up the longer lines into two shorter lines. Don't forget to leave room for the user's name in lines that include NA$.

One last item. When the computer says I OWE YOU 37 CENTS, it is speaking for the hardware owner, not the software author. In other words, the "I" ain't me; the "I" is you! Don't send me a bill stating that I owe your kid $87.52 for a job well done. Unleash this program on your kids at your wallet's peril!

Program 1: Color Computer, Apple, Commodore Version

```
120 REM CHR$(147)= CLEAR SCREEN
140 REM CHR$(18)= REVERSE VIDEO ON
160 REM CHR$(146)= REVERSE VIDEO OF
    F
500 PRINT CHR$(147) "MAKING CHANGE"
    : PRINT: PRINT
520 INPUT "PLEASE ENTER YOUR NAME";
     NA$
540 X%=100*RND(-RND(0))+1
560 PC=PC+1: RC=0: PRINT CHR$(147)
580 IF INT(PC/2)=(PC/2) THEN 2000: ~
   REM PROBLEM TYPE SELECTION
1000 REM GIVE CHANGE PROBLEM ROUTINE

1020 PRINT PC " " NA$ " GIVE ME "X%"
     CENTS.": PRINT
1040 INPUT "HOW MANY QUARTERS"; Q: Q
    1=Q*25: PRINT
1060 INPUT "HOW MANY DIMES"; D: D1=D
    *10: PRINT
1080 INPUT "HOW MANY NICKELS"; N: N1
    =N*5: PRINT
1100 INPUT "HOW MANY PENNIES"; P: PR
    INT: PRINT
1120 Q%=Q: D%=D: N%=N: P%=P: TC=Q1+D
    1+N1+P
1140 IF Q%<>Q OR D%<>D OR N%<>N OR P
    %<>P THEN GOSUB 5000: GOTO
     1220
1160 IF P=X% AND X%>4 AND TC=X% THEN
     GOSUB 6000: GOTO 1220
1180 IF X%=TC THEN 3000
1200 GOSUB 4000
1220 IF RC>3 THEN 8000
1240 GOTO 1020: REM REPEAT PROBLEM
2000 REM COUNT CHANGE PROBLEM ROUTIN
    E
2020 PRINT PC " " NA$ ", I HAVE:"
2040 XX%=X%: QU%=XX%/25: XX%=XX%-QU%
    *25
2060 DI%=XX%/10: XX%=XX%-DI%*10: NI%
    =XX%/5: PE%=XX%-NI%*5
2080 PRINT: PRINT TAB(10) QU% "QUART
    ERS,"
2100 PRINT: PRINT TAB(10) DI% "DIMES
    ,"
2120 PRINT: PRINT TAB(10) NI% "NICKE
    LS, AND"
2140 PRINT: PRINT TAB(10) PE% "PENNI
    ES."
2160 IF RC>3 THEN RETURN: REM FOR CO
    IN PRINT OUT AT 8040
2180 PRINT: INPUT "HOW MUCH CHANGE D
    O I HAVE"; TC: PRINT: PRIN
    T
2200 IF INT(TC)<>TC THEN PRINT NA$ "
    , DON'T USE DECIMAL POINTS
    .": GOTO 2180
2220 IF X%=TC THEN 3000
2240 GOSUB 4000
2260 IF RC>3 THEN 8000
2280 GOTO 2020: REM REPEAT PROBLEM
3000 REM CORRECT ANSWER ROUTINE *** ~
    LINES 3001 TO 3899 FOR USE
    R GRAPHICS.
3900 PRINT CHR$(18) "CORRECT " NA$ "
    !!!!!" CHR$(146)
3920 PRINT: PRINT "YOU EARN 1 CENT!!
    !!"
3940 ME=ME+1: GOTO 7000
4000 REM WRONG ANSWER ROUTINE
4020 IF TC>X% THEN 4060
4040 PRINT X%-TC "CENTS SHORT " NA$ ~
    "!": GOTO 4080
```

AARDVARK

TRS-80 COLOR OSI VIC-64 VIC-20 SINCLAIR TIMEX

TUBE FRENZY
(by Dave Edson)
This is an almost indescribably fast action arcade game. It has fast action, an all new concept in play, simple rules, and 63 levels of difficulty. All machine code, requires Joysticks. Another great game by Dave Edson. TRS 80 COLOR ONLY. 16k and Joysticks required. $19.95.

QUEST – A NEW IDEA IN ADVENTURE GAMES! Different from all the others. Quest is played on a computer generated map of Alesia. Your job is to gather men and supplies by combat, bargaining, exploration of ruins and temples and outright banditry. When your force is strong enough, you attack the Citadel of Moorlock in a life or death battle to the finish. Playable in 2 to 5 hours, this one is different every time. 16k TRS-80, TRS-80 Color, and Sinclair. 13K VIC-20. $14.95 each.

ADVENTURES!!!
These Adventures are written in BASIC, are full featured, fast action, full plotted adventures that take 30-50 hours to play. (Adventures are interactive fantasies. It's like reading a book except that you are the main character as you give the computer commands like "Look in the Coffin" and "Light the torch.")

Adventures require 16k on TRS80, TRS80 color, and Sinclair. They require 8k on OSI and 13k on Vic-20. Derelict takes 12k on OSI. $14.95 each.

CATERPILLAR
O.K., the Caterpillar does look a lot like a Centipede. We have spiders, falling fleas, monsters traipsing across the screen, poison mushrooms, and a lot of other familiar stuff. COLOR 80 requires 16k and Joysticks. This is Edson's best game to date. $19.95 for TRS 80 COLOR.

PROGRAMMERS!
SEE YOUR PROGRAM IN THIS SPACE!!
Aardvark traditionally pays the highest commissions in the industry and gives programs the widest possible coverage. Quality is the keyword. If your program is good and you want it presented by the best, send it to Aardvark.

ESCAPE FROM MARS
(by Rodger Olsen)
This ADVENTURE takes place on the RED PLANET. You'll have to explore a Martian city and deal with possibly hostile aliens to survive this one. A good first adventure.

PYRAMID (by Rodger Olsen)
This is our most challenging ADVENTURE. It is a treasure hunt in a pyramid full of problems. Exciting and tough!

HAUNTED HOUSE (by Bob Anderson)
It's a real adventure—with ghosts and ghouls and goblins and treasures and problems — but it is for kids. Designed for the 8 to 12 year old population and those who haven't tried Adventure before and want to start out real easy.

DERELICT
(by Rodger Olsen & Bob Anderson)
New winner in the toughest adventure from Aardvark sweepstakes. This one takes place on an alien ship that has been deserted for a thousand years — and is still dangerous!

CATCH'EM
(by Dave Edson)
One of our simplest, fastest, funnest, all machine code arcade games. Raindrops and an incredibe variety of other things come falling down on your head. Use the Joysticks to Catch'em. It's a BALL! — and a flying saucer! — and a Flying Y!— and so on. TRS 80 COLOR. $19.95.

BASIC THAT ZOOOMMS!!
AT LAST AN AFFORDABLE COMPILER!
The compiler allows you to write your programs in easy BASIC and then automatically generates a machine code equivalent that runs 50 to 150 times faster.

It does have some limitations. It takes at least 8k of RAM to run the compiler and it does only support a subset of BASIC—about 20 commands including FOR, NEXT, END, GOSUB, GOTO, IF, THEN, RETURN, END, PRINT, STOP, USR (X), PEEK, POKE, *, /, +, -, > , < , =, VARIABLE NAMES A-Z, SUBSCRIPTED VARIABLES, and INTEGER NUMBERS FORM 0-64K.

TINY COMPILER is written in BASIC. It generates native, relocatable 6502 or 6809 code. It comes with a 20-page manual and can be modified or augmented by the user. $24.95 on tape or disk for OSI, TRS-80 Color, or VIC.

Please specify system on all orders

ALSO FROM AARDVARK – This is only a partial list of what we carry. We have a lot of other games (particularly for the TRS-80 Color and OSI), business programs, blank tapes and disks and hardware. Send $1.00 for our complete catalog.

AARDVARK - 80
2352 S. Commerce, Walled Lake, MI 48088
(313) 669-3110

Phone Orders Accepted 8:00 a.m. to 4:00 p.m. EST. Mon.-Fri.

```
4060 PRINT TC-X% "CENTS TOO MUCH " N
    A$ "!"
4080 PRINT: PRINT "YOU LOSE 1 CENT!"

4100 ML=ML+1
4120 RC=RC+1: IF RC>3 THEN RETURN
4140 PRINT: PRINT "TRY AGAIN " NA$ "
    .": PRINT: PRINT
4160 IF RC=1 THEN W=W+1
4180 RETURN
5000 REM FRACTIONAL ANSWER ROUTINE
5020 PRINT NA$ ", ANSWERS WITH DECIM
    AL"
5040 PRINT: PRINT "POINTS ARE NOT AL
    LOWED."
5060 PRINT: PRINT "YOU ARE FINED 1 C
    ENT."
5080 F=F+1: GOTO 4120
6000 REM ALL PENNY ANSWER
6020 PRINT NA$ "," P "PENNIES DOES E
    QUAL" X% "CENTS."
6040 PRINT: PRINT "BUT THAT IS CHEAT
    ING."
6060 PRINT: PRINT "YOU ARE FINED 1 C
    ENT FOR CHEATING."
6080 F=F+1: GOTO 4120
7000 REM SCORING ROUTINE
7020 PRINT: PRINT: PRINT "PRESS " CH
    R$(18) "S" CHR$(146) " FOR
     SCORE,";
7040 PRINT " ANY KEY TO CONTINUE."
7060 FOR X=1 TO 10: GET A$: NEXT X
7080 GET A$: IF A$="" THEN 7080
7100 IF A$<>"S" THEN 540
7120 PRINT CHR$(147) NA$ "'S SCORE!!
    "
7140 PRINT: PRINT "TOTAL PROBLEMS:" ~
    PC
7160 PRINT: PRINT "TOTAL CORRECT:" P
    C-W
7180 PRINT: PRINT "TOTAL WRONG:" W
7200 PRINT: PRINT "PERCENT CORRECT:"
     (PC-W)/PC*100 "%"
7220 PRINT: PRINT "MONEY EARNED:" ME
     "CENTS"
7240 PRINT: PRINT "MONEY LOST:" ML "
    CENTS"
7260 PRINT: PRINT "FINES:" F "CENTS"
    : PRINT
7280 PA=ME-ML-F
7300 IF PA<0 THEN PA=ABS(PA): PRINT ~
    "YOU OWE ME" PA "CENTS!": ~
    GOTO 7340
7320 PRINT "I OWE YOU" PA "CENTS!"
7340 GOTO 7000
8000 REM CORRECT ANSWER PRINT OUT
8020 PRINT: PRINT: PRINT NA$ ", THE ~
    CORRECT ANSWER IS:"
8040 GOSUB 2040: REM COIN PRINT OUT
8060 PRINT: PRINT " MAKES" X% "CENTS
    ."
8080 GOTO 7000
```

Program 2: Atari Version

```
20 REM MAKING CHANGE  ATARI VERSION
30 DIM NA$(30)
100 REM SCREEN ASCII CODES
120 REM CHR$(125)= CLEAR SCREEN
500 PRINT CHR$(125);"MAKING CHANGE":P
    RINT :PRINT
520 PRINT "PLEASE ENTER YOUR NAME";:I
    NPUT NA$
540 X=INT(100*RND(-RND(0))+1)
560 PC=PC+1:RC=0:PRINT CHR$(125)
580 IF INT(PC/2)=(PC/2) THEN 2000:REM
     PROBLEM TYPE SELECTION
1000 REM GIVE CHANGE PROBLEM ROUTINE
1020 PRINT PC;") ";NA$;", GIVE ME ";X
     ;" CENTS.":PRINT
1040 PRINT "HOW MANY QUARTERS";:INPUT
     Q:Q1=Q*25:PRINT
1060 PRINT "HOW MANY DIMES";:INPUT D:
     D1=D*10:PRINT
1080 PRINT "HOW MANY NICKELS";:INPUT
     N:N1=N*5:PRINT
1100 PRINT "HOW MANY PENNIES";:INPUT
     P:PRINT :PRINT
1120 IF Q<>INT(Q) OR D<>INT(D) OR N<>
     INT(N) OR P<>INT(P) THEN GOSUB 5
     000:GOTO 1220
1140 Q=INT(Q):D=INT(D):N=INT(N):P=INT
     (P):TC=Q1+D1+N1+P
1160 IF P=X AND X>4 AND TC=X THEN GOS
     UB 6000:GOTO 1220
1180 IF X=TC THEN 3000
1200 GOSUB 4000
1220 IF RC>3 THEN 8000
1240 GOTO 1020:REM REPEAT PROBLEM
2000 REM COUNT CHANGE PROBLEM ROUTINE
2020 PRINT PC;") ";NA$;", I HAVE:"
2040 XX=X:QU=INT(XX/25):XX=XX-QU*25
2060 DI=INT(XX/10):XX=XX-DI*10:NI=INT
     (XX/5):PE=XX-NI*5
2080 PRINT :PRINT QU;" QUARTERS,"
2100 PRINT :PRINT DI;" DIMES,"
2120 PRINT :PRINT NI;" NICKELS, AND"
2140 PRINT :PRINT PE;" PENNIES."
2160 IF RC>3 THEN RETURN :REM FOR COI
     N PRINT OUT AT 8040
2180 PRINT :PRINT "HOW MUCH CHANGE DO
      I HAVE";:INPUT TC:PRINT :PRINT
2200 IF INT(TC)<>TC THEN PRINT NA$;",
      DON'T USE DECIMAL POINTS.":GOTO
      2180
2220 IF X=TC THEN 3000
2240 GOSUB 4000
2260 IF RC>3 THEN 8000
2280 GOTO 2020:REM REPEAT PROBLEM
3000 REM CORRECT ANSWER ROUTINE *** L
     INES 3001 TO 3899 FOR USER GRAPH
     ICS.
3900 PRINT "CORRECT ";NA$;"!!!!!"
3920 PRINT :PRINT "YOU EARN 1 CENT!!!
     !"
3940 ME=ME+1:GOTO 7000
4000 REM WRONG ANSWER ROUTINE
4020 IF TC>X THEN 4060
4040 PRINT X-TC;" CENTS SHORT ";NA$;"
     !":GOTO 4080
4060 PRINT TC-X;" CENTS TOO MUCH ";NA
     $;"!"
4080 PRINT :PRINT "YOU LOSE 1 CENT!"
4100 ML=ML+1
4120 RC=RC+1:IF RC>3 THEN RETURN
4140 PRINT :PRINT "TRY AGAIN ";NA$;".
```

```
      ":PRINT :PRINT
4160 IF RC=1 THEN W=W+1
4180 RETURN
5000 REM FRACTIONAL ANSWER ROUTINE
5020 PRINT NA$;", ANSWERS WITH DECIMA
     L"
5040 PRINT :PRINT "POINTS ARE NOT ALL
     OWED."
5060 PRINT :PRINT "YOU ARE FINED 1 CE
     NT."
5080 F=F+1:GOTO 4120
6000 REM ALL PENNY ANSWER
6020 PRINT NA$;",";P;"PENNIES DOES EQ
     UAL";X;"CENTS."
6040 PRINT :PRINT "BUT THAT IS CHEATI
     NG."
6060 PRINT :PRINT "YOU ARE FINED 1 CE
     NT FOR CHEATING."
6080 F=F+1:GOTO 4120
7000 REM SCORING ROUTINE
7020 PRINT :PRINT :PRINT "PRESS S FOR
      SCORE,"
7040 PRINT " ANY KEY TO CONTINUE."
7060 POKE 764,255
7080 K=PEEK(764):IF K=255 THEN 7080
7100 POKE 764,255:IF K<>62 THEN 540
7120 PRINT CHR$(125);NA$;"'S SCORE!!"
7140 PRINT :PRINT "TOTAL PROBLEMS:";P
     C
7160 PRINT :PRINT "TOTAL CORRECT:";PC
     -W
7180 PRINT :PRINT "TOTAL WRONG:";W
7200 PRINT :PRINT "PERCENT CORRECT:";
     INT((PC-W)/PC*100);"%"
7220 PRINT :PRINT "MONEY EARNED:";ME;
     " CENTS"
7240 PRINT :PRINT "MONEY LOST:";ML;"
     CENTS"
7260 PRINT :PRINT "FINES:";F;" CENTS"
     :PRINT
7280 PA=ME-ML-F
7300 IF PA<0 THEN PA=ABS(PA):PRINT "Y
     OU OWE ME ";PA;" CENTS!":GOTO 73
     40
7320 PRINT "I OWE YOU ";PA;" CENTS!"
7340 GOTO 7000
8000 REM CORRECT ANSWER PRINT OUT
8020 PRINT :PRINT :PRINT NA$;", THE C
     ORRECT ANSWER IS:"
8040 GOSUB 2040:REM COIN PRINT OUT
8060 PRINT :PRINT " MAKES";X;"CENTS."
8080 GOTO 7000
```

©

Learning With Computers

Glenn M. Kleiman

Each of the prior *Learning with Computers* columns focused on a single main topic. Word processing, graphics, Logo, PILOT, preschool computing, computer camps, introducing programming, computer literacy and other topics have been discussed in the past year. Meanwhile, software, books, comments from readers, and other pieces of information have been piling up on my desk, waiting to be fit into a column. For the next few months, I will work my way through this stack, and not focus on any particular topics.

Information Directories

The stacks on my desk illustrate that products, information, resource centers, and uses of computers in education are increasing at an astounding rate. Fortunately, there are people who collect and organize this information for the convenience of the rest of us. You may find the following recently published directories useful:

Instructor Magazine Computer Directory For Schools. This directory contains four main categories of information: hardware, software, publications, and companies. Hardware is subdivided into computers, memory devices, monitors, printers, modems, networking devices, graphics devices, and other categories. Software is listed by area of use – language arts, mathematics, science, social studies, art, music, and so on.

Publications are divided into books, magazines, and other resources. For each of the nearly 2,000 products listed, there is a brief description, including, when relevant, equipment requirements, grade level, source, and price. There are also 400 companies listed. The aim of this directory is to be encyclopedic rather than selective, so no evaluative information is provided. This directory is available for $19.95 (plus $2 shipping) from Instructor Books, P.O. Box 6177, Duluth, MN 55806.

Classroom Computer News Directory. Part I of this directory lists and describes sources of information. It is divided into six sections. The first lists and briefly describes anthologies, bibliographies, indexes, on-line data bases, resource centers, and research and development projects. The second section covers software directories, sources of reviews, and clearing-houses. The remaining sections cover associations, periodicals, funding, and miscellaneous resources.

Part II is divided according to computer systems, with sections covering Apple, Atari, Commodore, Radio Shack, Sinclair, and Texas Instruments. Periodicals, software directories, and user groups are listed for each computer.

Part III covers local and regional resources, organized by state and province. User groups, projects, organizations, computer learning centers, and state or provincial personnel responsible for educational computing are listed.

Part IV lists colleges and universities offering courses on computers in education, and the types of courses offered. Part V is a calendar of national and regional conferences and workshops, and Part VI is a "yellow pages" of paid advertisements.

Overall, the directory contains a great deal of well organized information which can help you find answers to all sorts of questions. It is available for $14.95 from *Classroom Computer News*, 341 Mt. Auburn Street, Watertown, MA 02172.

Microcomputer Directory: Applications In Educational Settings. This directory lists projects using microcomputers for instructional and administrative purposes at over 1,000 sites in the United States. It includes elementary and secondary schools, computer camps, museums, prisons, alternative learning sites, and colleges and universities. Each listing includes a brief description of the project and the name and address of a person to contact for more information. It is available for $15 (plus $1 postage) from Gutman Library, Harvard University Graduate School of Education, Appian Way, Cambridge, MA 02138.

On Software Reviewing

During the past year, I have discussed many software packages. As some readers have noted, my overall evaluations have generally been very positive. There is a simple reason for this. I try to bring useful, innovative, well-designed software to your attention, and I do not take the time or space to mention software I find lacking or uninteresting. That is, I filter poor software rather than criticize it. Of course, this doesn't mean that I have found any perfect programs, but there certainly are some very good ones available now.

More Programs For Preschoolers

In last May's column, I discussed some principles

UNIQUE MULTI-USER SOFTWARE BRINGS NEW EXCITEMENT TO GROUP LEARNING.

The results are always the same. Put a computer in a classroom and children are drawn to it like steel to a magnet. And even though only one child actually uses the computer, the others coach or offer encouragement. Involving as this activity may be, it fails to take advantage of one of the best known principles of learning. But more about this later.

A simple idea.

When two educational researchers, Dr. Matilda Butler and Dr. William Paisley, studied the interaction of children around microcomputers they had an interesting, yet simple, idea. Instead of one user and several observers, why not give every child the opportunity to learn simultaneously. This idea sparked an entire line of unique educational software and gave birth to a new company, Edupro.

Learning through cooperation and competition.

Each one of Edupro's Microgroup™ computer programs presents your students with a different learning environment. It may be a visit with storybook friends. A trip through American history. Or an exploration of the world around us.

In any case, the principles are the same. Mathematical, language arts, social studies, and science problems are presented as contests, races, and puzzles. Using joysticks or paddles up to eight children work together, either competitively or cooperatively. They race against time, each other, or both.

Forgotten principle.

Now about that principle of learning other educational software ignores.

For years, studies have shown that children learn more efficiently in groups. Group learning motivates slower learners to persevere. It promotes divergent thinking. And it teaches the importance of working together for a common goal.

Atari® and 400/800™ are trademarks of ATARI Inc.

Ordinary educational software can't provide this stimulation. But with Edupro software children can experience the challenge and excitement of group learning on a daily basis.

Designed for the simplest computers.

Even with all the advances in computer science and micro-electronics, multi-user software typically requires a sophisticated, expensive computer. At a cost beyond the reach of most school districts. So the following paragraphs may contain the best news of all.

These unique programs run on Atari 400 or Atari 800 personal computers. They're available on floppy disk or cassette, and use the minimum amount of computer memory (16K bytes). So even the simplest Atari computer can teach eight students simultaneously.

And the learning doesn't have to stop in your classroom.

These Atari units are also one of the most popular home computers, so Edupro programs can involve the entire family in the group learning process. Not only can parents work with their children, brothers and sisters can share learning with each other. A feat that's hard to duplicate inside a classroom.

Your own hands-on experience.

If you were at this fall's Computer-Using Educators Conference you may have had a demonstration of our programs. Hundreds of educators did. Many of them said that this was an effective way to judge the potential of these programs. But you can have a better opportunity.

We've prepared a sampler kit of the conferences' most popular four user programs. It includes selections from six different programs spanning ages five to adult (all our programs are age graded). We'll be happy to send it to you so you can introduce these programs to your own students. The kit comes with complete instructions and our catalog listing over

50 additional programs. Plus we'll include a coupon good for a 10% discount on your first order.

We know of no other software that can turn a microcomputer into a tool for sharing the excitement of group learning.

Fill out the order form below and see the results in your own classroom.

I want to share the excitement of group learning with my students. Please send me the number of sampler kits I've indicated below. I understand that each kit includes a disk or cassette (my choice) of selected Edupro programs, instructions, catalog, and 10% discount coupon for my next order.

_____ Sampler kit(s) with disk @ $7.95 each _____

_____ Sampler kit(s) with cassette @ $7.95 each _____

California residents add sales tax _____

First Class postage & handling $ 2.00

Total: Check or money order enclosed for $ _____

Please bill ____MasterCard ____Visa

(card no.) (exp. date)

Name _____

Address _____

City _____

State _____ ZIP _____

Signature _____

Allow 3 weeks for delivery.

send to: Edupro, Dept. CO, P.O. Box 51346, Palo Alto, California 94303

Edupro™

of designing software for young children and described seven educational programs suitable for preschoolers. Since then, I have received several new excellent programs for young children.

Facemaker combines a creative tool with a memory game. The child begins by creating a face. The program presents sets of mouths, noses, ears, eyes, and hair. The child selects the specific parts, and the computer automatically combines them into a face. *Facemaker* will be reviewed in an upcoming issue.

My First Alphabet, by Fernando Herrara, is designed to help children learn to recognize letters and numbers, associate words with each letter, count, and learn to use the computer keyboard. This program was the first-place winner of an Atari competition, and is marketed by Atari.

For each letter of the alphabet, the program contains a screen display with a picture, a large version of the letter, and four words that begin with it. For example, for the letter A, first an airplane and a large A are drawn on the screen. Then large letters show the sentence "A IS FOR AIRPLANE," and the words ARROW, ARM and ANT appear. Each number also has its own display. For example, for the number five, nine clowns are drawn on the screen, then smiling faces are added to five of them and frowning faces are added to the other four.

After an initial display of the letters, accompanied by the alphabet song, the program provides several ways of using the screen displays. You can have the computer automatically show the displays for randomly selected letters and numbers, or you can specify which displays are to be shown. You can have the displays run continuously, or have the program wait for the child to type the matching letter or number for each screen display. When the child types a correct key, music plays and the screen colors change.

The pictures are excellent and realistic, so children can easily recognize each item portrayed. The program holds children's attention, and they enjoy seeing the pictures, recognizing the letters, and finding the right keys to press.

My First Alphabet is designed for a child to use with an adult present to exchange questions and answers, prompt the child, read the words, and so on. This is important since the program itself requires very little activity on the child's part. It takes quite awhile to draw each screen display and, once the display is completed, the most the child is to do is to press the corresponding key. Without someone to question, prompt and guide them, children using this program spend most of their time watching, and very little time doing anything.

Children's Television Workshop Software

Children's Television Workshop (CTW), producers of *Sesame Street, The Electric Company* and *3-2-1 Contact*, has developed four disks of play-and-learn programs for Apple II computers. Each disk contains four programs, many of which are variations of classic games and puzzles, such as *Hangman, Anagrams,* and *Tower of Hanoi*. Less expensive computer versions of these games and puzzles are available elsewhere, such as from user groups or from program listings in books and magazines.

However, the CTW versions are especially well-designed, easy to use, and contain fine graphics. There are disks for children four to seven years old, seven to ten years old, nine to thirteen years old, and a disk containing programs suitable for a wide range of ages. Each disk comes with a book containing instructions for the programs and suggestions for related non-computer activities and games.

Ernie's Quiz is the disk for the youngest children. One program on it, called "Guess Who," uses colorful, low-resolution pictures of Sesame Street Muppets, such as Bert, Ernie, Big Bird, and the Cookie Monster. The computer begins by displaying a few blocks of color and then it gradually fills in more and more of the picture. The child tries to figure out, as quickly as possible, which Sesame Street character is being portrayed.

A "Face-It" program lets children create faces on the screen. It is similar in concept to the *Facemaker* program described above, but Face-It has larger, more colorful faces, lets the child add more features (such as eyeglasses, mustaches, beards), and lets the child control colors. The game paddles are used to select features and colors. (Face-It does not contain the animation and memory game options found in *Facemaker*.)

There are two more programs on the disk. "Jelly Beans" is a simple counting game. The "Ernie's Quiz" program provides hints about a Muppet, and the child is to choose which Muppet is being described.

Mix and Match is the disk for all ages. One program on this disk lets children create pictures by combining the heads, bodies, and feet of different Muppets. For example, the child can create Muppet with Bert's head, Big Bird's body, and Oscar the Grouch's feet. The computer will show the picture and tell the children that this creature is called "Berber the Grouch."

The disk also contains an excellent version of *Animal*, a classic computer game in which the player thinks of an animal and the computer tries to figure out which animal it is by asking questions such as "Does it live on land?" and "Does it fly?" If the computer cannot guess the animal, it asks the player questions. In this way, the computer adds information to its knowledge base so it improves how well it plays the game.

There are two other programs. "Layer Cake" is a version of the *Tower of Hanoi* puzzle. It is easy to use and takes good advantage of the Apple's graphics capabilities. "Raise the Flags" is a non-violent variation of *Hangman*. The disk also contains a word editor which lets you enter your own words for the Raise the Flags program.

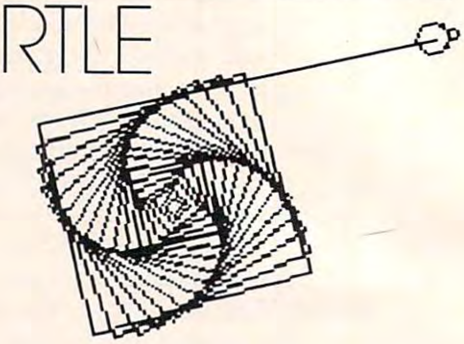
Instant Zoo is a disk for children ages seven to ten. Its four games are: Instant Zoo, a picture recognition game similar to Guess Who, but with animals instead of Muppets; Star Watch, which measures how long the child takes to press a key after a shooting star appears on the screen; Quick Match, in which two words are shown and the child presses one key if they match and another if they do not match; and Scramble, an anagram game in which the child races the computer to unscramble words. There is also a word editor for creating your own word lists for Quick Match and Scramble.

Spotlight is a disk for children ages nine to thirteen. It has two programs in which the child turns mirrors to direct lights to targets. The third program, called "Hot Stuff," is a game of logic in which the player tries to guess the computer's secret three-digit number. After each guess, the computer tells how many of the three digits are in the right place, and how many appear in the secret number, but not in the same place as guessed. Sounds simple, but complex logic is needed to figure out the secret number with as few guesses as possible.

The CTW disks are marketed by Apple and are available from Apple dealers. Each disk costs $50. The Mix and Match disk programs are in Applesoft. The programs on the other disks are in Integer BASIC, so they require either an Apple computer with Integer BASIC, or one with 64K (in which case Integer BASIC will automatically load into the extra memory when the disk is booted). Some of the programs also require game paddles.

The CTW programs, *Facemaker*, *My First Alphabet*, and those I reviewed in last May's column provide an excellent and varied set of software for introducing young children to computers. ©

FRIENDS OF THE TURTLE



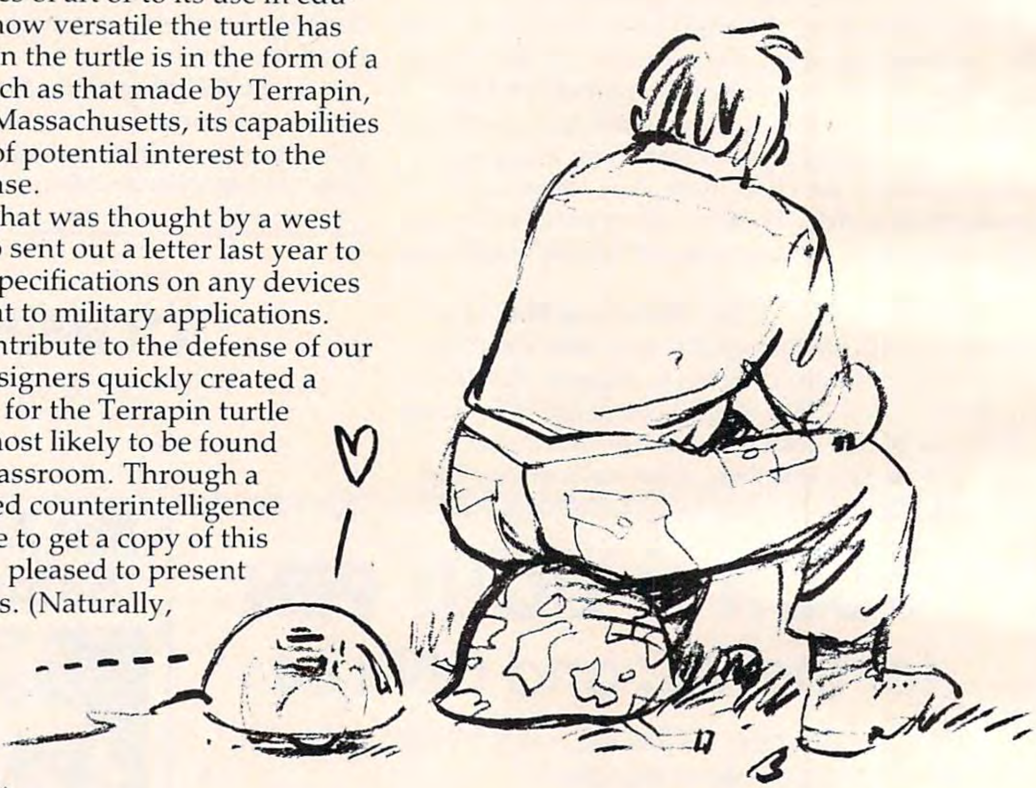
David D. Thornburg, Associate Editor

The Department Of Turtle Defense

Those of us who limit our use of turtle graphics to the aesthetic pleasures of art or to its use in education have no idea how versatile the turtle has become. In fact, when the turtle is in the form of a mechanical robot, such as that made by Terrapin, Inc., in Cambridge, Massachusetts, its capabilities are so great as to be of potential interest to the Department of Defense.

At least this is what was thought by a west coast think-tank who sent out a letter last year to Terrapin asking for specifications on any devices that might be relevant to military applications.

Ever eager to contribute to the defense of our country, Terrapin designers quickly created a military specification for the Terrapin turtle – a $600 peripheral most likely to be found in a primary grade classroom. Through a network of well placed counterintelligence operatives, I was able to get a copy of this specification and am pleased to present the following excerpts. (Naturally, I have made sure that I haven't included any information that would compromise our nation's defense.)



From a functional viewpoint, Turtles show great promise as all-terrain vehicles for pushing heavy payloads to their destination. Under the heading of survivability, we find that:

> *The turtle enjoys a low observability, due to a minimal radar cross section and an almost non-existent infrared signature. In addition, its ground-hugging characteristics maximize terrain masking, resulting in lower target acquisition by most classes of SSM and ASM threats. ... The Turtle can make a 180-degree turn in less space than any military vehicle currently in use by US forces, ground, air, or sea. With minor modifications, a Turtle could be constructed that could double its cruising speed for a terminal "dash" capability that would greatly enhance survivability in the endgame.*
>
> *... Even if a suitable counter were found to all these properties of the Turtle, it is doubtful that an enemy could afford to deploy counter-weapons in sufficient number to nullify the possibility of defense saturation in the event of an all-out Turtle attack.*

On the topic of range ...

> *The Mark I, Mod 0 Turtle has an effective range of some 3 to 4 meters, depending on its winding count. Range is most severely limited by the Turtle's cable, but this limitation is trivial by comparison to the inherent advantages of wire-guidance. ... Furthermore, our research department is currently engaged in the testing of a 100 mile cable for the Turtle. ... While this does result in a shorter tooth-to-tail ratio, we feel it could significantly enhance the battlefield capabilities of the Turtle installations.*

On the topic of guidance ...

> *Because the Terrapin Turtle™ is computer con-*

trolled, military data processing technicians can write arbitrarily baroque programs that will cause it to do pretty much unpredictable things. Even if an enemy had access to the programs that guided the Turtle Task Team™, it is quite likely that they would find them impossible to understand, especially if they were written in ADA. In addition, with judicious use of the Turtle's touch sensors, one could, theoretically, program a large group of Turtles to simulate Brownian motion. The enemy would hardly attempt to predict the paths of some 10,000 Turtles bumping into each other more or less randomly on their way to performing their mission. Furthermore, we believe that the spectacle would have a demoralizing effect on enemy ground troops.

And what about munitions?

The Terrapin Turtle™ does not currently incorporate any munitions, but even civilian versions have downward-defense capability. The Turtle can be programmed to attempt to run over enemy forces on recognizing them, and by raising and lowering its pens at about 10 cycles per second, puncture them to death.

Turtles can be easily programmed to push objects in a preferred direction. Given this capability, one can easily envision a turtle discreetly nudging a hand grenade into an enemy camp, and then accelerating quickly away.

But what does it cost to install one?

The Terrapin Turtle is designed for installation at no cost by children and elementary school teachers. We feel that the military installation cost should be under $10,000 per unit.

I can think of no greater deterrent to all-out war than masses of robot turtles landing on the beaches and steadily moving towards the enemy.

Think of the tremendous opportunities for new patriotic songs: "When 4XQ7 Comes Crawling Home Again (Ta Raa, Ta Raa)," "Over There (Forward 20, Right 30), Over There," "How Are You Going to Keep Them Running POLY After They've Seen Paree?"

Instead of Basic training, the turtles will, no doubt, have to go through Logo training with procedures such as:

```
TO HUP :NUM1 :NUM2 :NUM3
REPEAT 4 [FD 10 WAIT 20]
HUP 2 3 4
END
```

Hats off to the Terrapin Patriots! May this be an ever safer world for turtles. ©

Recreational Computing Back Issues

Recreational Computing was the first and only personal computing magazine when it started in 1972 (it was called the *PCC Newspaper* back then). Bob Albrecht, David Thornburg, Isaac Asimov, Don Inman, Ramon Zamora, Robert Jastrow, Mac Oglesby, Adam Osborne – the list of authors reads like a Who's Who of microcomputing. These and many other authors contributed some of the finest articles about computers and now-classic games to the pages of *Recreational Computing*.

Last fall, *Recreational Computing* was merged into **COMPUTE!** and we are now offering available back issues. Whatever your interest, you'll find something here – from Spanish BASIC to Computers in Sports Medicine, from Future Fantasy Games to Robot Pets.

September 1974 A Practical, Low-cost Home/School Microprocessor System, The Computer Illiteracy Problem, Eight Games In BASIC

March 1975 Build Your Own BASIC, The Computer In Art, Biorhythms

March/April 1976 A TTY Game, Games With The Pocket Calculator, Dodgem, Square, Tiny BASIC To Go

July 1976 BASIC Music, Tiny Trek For Altair, 16 Bit Computer Kit, Musical Numbers Guessing Game, Programmer's Toolbox

September/October 1976 Computer Games In The Classroom, Planets Game, Dungeons And Dragons, Hats Game, Pythagoras And Rational Music

November/December 1976 Story, Snake, Packl, Frogs Games, Make Believe Computers, The First West Coast Computer Faire, Subroutines, The First Computer

January/February 1977 Robot Pets, Computers And Space, Tiny Languages, Teaching Using Conversational Programming, High School Computers, Reverse, Tiny PILOT, Mastermind

March/April 1977 Z-80 PILOT, 6502 Assembly Programming, Tiny BASIC For Beginners, Math Drills & Games, Community Information Systems, Mine, Sales Simulation, Native American Board Games

July/August 1977 Do-it-yourself CAI, Pet Robots: New Capabilities, PILOT, CAI In BASIC, Programming The HP-25, Capture, Inverse Reverse, 8080 Matrix Subroutines, Women And Computers

September/October 1977 The $595 PET, More Tiny Languages, Computer Networks, The Bead Game, Biofeedback And Microcomputers Part 1, Home Energy Management, Sandpile Game, A BASIC PILOT

January/February 1978 Pascal Vs. BASIC, COMAL: Structured BASIC, Video Disks: Magic Lamps for Educators?, A Computer Revolution?, Pounce, The Mechanics of Robots, TRS-80: A Status Report

March/April 1978 Epic Computer Games, Micros for the Handicapped, Buckets Game, Prayer Wheel Program, Computer Contagion, Measuring Time, Frog Race, The IBM 370 Model 69

July/August 1978 Computer Whiz Kids, Public Access To Computers, Man-made Minds, Post-human Intelligence, A Modern-day Medicine Show, Live Wire Design, ASCII Graphix, Baseball, Concentration, Gambler's Paradox

September/October 1978 Kingdom Game, Computers and Museums, Sorcerer of Exidy, Snooping With Your PET, APL, Decimals in Tiny BASIC, Apple Math, TRS-80 Level II: A Grown-up Field Evaluation

November/December 1978 APL Games, The Return of the Dragons, Animated Games for TRS-80, Runequest, All In The Mind, The L-5 Society, Phantnum, Some Guidelines for Microcomputer Chess, Dataman

January/February 1979 A Jules Verne Fantasy, Artificial Intelligence, The Apple Corps is With Us, TRS-80 Personal Software, Vending Machine Gets "Brain," Apple II I/O, The Memory Game, REINO: Spanish Kingdom

March/April 1979 Calculator Comics, "Lord of the Rings," Chess Reconsidered, Database: Beastiary, Color Your Own Graphics, Universe, Easy POKEing with Applesoft BASIC, Air Raid, TRS-80 3-D Plots, Slot, Apple Rose

May/June 1979 PILOT for Apple II, The Game of Life, Gold Handicapping, Hunt, BASIC vs. Pascal, Inspector Clew-so, Flash for SOL, Faster Jumble, Concept Sans Computer, A Beginner's Guide To FRP

July/August 1979 Summer Fun, Fooling Around With Your PET, Cryptarithms, Baseball, Newett Awl's Goat, Zork: A Computerized Fantasy Simulation Game, What Light on Yonder Panel Flashes, The Dedicated Word Processor, The FORTE Music Programming Language

September/October 1979 TRS-80: Outside Connection, The Architecture of Multi-Player Games, The Sounds of Texas Instruments, Dynamic Color Graphics on the New Atari, An Apple PILOT, Gandalf, Spanish BASIC, Designing Animal Games, APL Mastermind

November/December 1979 SHOGI: Games For You To Program, Atari Sounds, Texas Instrument Graphics and Animation, Interrupt, Match Me, Calendar, Making Music on the PET, Tower of Hanoi, Bingo, Animal Games

January/February 1980 Computing and Holistic Health, TI Graphics and Animation Part 2, Games To Program, New Directions in Numerical Computing, An Extended BASIC "IF" Facility, Beating Computer Anxiety, Capture for PET, 8080 Tic Tac Toe, Chainwalk, Programming Problems

March/April 1980 Special Games Issue: Recreation Apple II Hi-res Graphics, Delicious Functions, Galaxy II, Fairy Chess, Raging Robots, Program Instruction Builder, Data Retrieval: An Introduction

May/June 1980 Introduction to Computer Music, CBBS Phone Numbers, 6502 Machine Language, The Electric Phone Book, Number Translation, Sea Search, Apple Animation, Twister Move Generator, DOZO, Shell Game, Home Video Displays, A Proposed Graphics Language

July/August 1980 Fantasy Games Issue: Write Your Own Computer Fantasy Simulation, Wizard's Castle, On Future Fantasy Games, Wonderful World of Eamon, In Defense of Hackers, Touch Panels and Interactive Graphics

September/October 1980 Probability Trees: Big Business on the Micro, The Best of People's Computer Company, Computer Analysis of Athletics, Word Search, Computers in Sports Medicine, Wired, Revolution in Typography?, Textrapolation

November/December 1980 Computerized Voting, Computer-Using Educators, Hot-rod Computers, House of the Future, Yote, DOZO in Pascal, What is Truth?, Sixth Order Magic Squares on a TRS-80

January/February 1981 The Education Revolution: Computer Games in the Classroom, An Art-producing Turtle, Computer Literacy Resources, Musical Compositions Using Computers, Microcomputers in China, Twenty Questions, The Pirate's Life for Me, Computers and the Volcanic Fallout

March/April 1981 Space Exploration: Frontiers for You and Your Micro, Voyage To Antares: A Spaceship Simulator, The Computer as Chess Ally, Star Trek – A Dialogue Approach, Mark of Breeding (fiction), The Fifteen Puzzle

May/June 1981 Using Computers at Sesame Place, Atari PILOT and Turtle Graphics, Computer Anatomy for Beginners, The Impact of Micros, Nevada-style 8-spot Keno, Sketch Pad, Sum of the Digits, TRS-80 Property Management Program, The Pocket Corner

July/August 1981 Which Computer Should You Buy?, Commodore's New Rainbow Machine, The Wired Nation: Do We Want It?, Computers at the Junior Museum, 3-D Tic Tac Toe for PET, Number Crossword for all Computers

September/October 1981 43 Ways To Make Money With Your Micro, How To Start A Software Exchange, Who Are Computer Criminals?, Micros Behind Bars, Number Systems, Computer Knock-knock Jokes in BASIC and LISP, For Photographers Only, Fibonacci Nim, Roman Numeral Conversum Programum

Special *Recreational Computing* Back Issue Pricing:

Single Issue:	**$ 3.00**	**Any Fifteen Issues:**	**$25.00**
Any Five Issues:	**$10.00**	**Any Twenty Issues:**	**$30.00**
Any Ten Issues:	**$20.00**		

For Fastest Service, Call Toll Free **800-334-0868**
In NC Call **919-275-9809**

Or Send Order and Payment to **COMPUTE!** Publications, P.O. Box 5406, Greensboro, NC 27403.

In the US, please include $.20 per issue ordered for shipping and handling. Outside the US, please include $.30 per issue for surface mail. Orders must be prepaid in US funds or international money order. All orders subject to availability.

A Computer Language For Kids

Fred D'Ignazio, Associate Editor

In this column we have explored ways to make computers more accessible to kids. In the August to December columns, for example, we developed a computer "friend" for kids.

When the friend program is run, the friend's face appears on the TV screen. At first, the friend is asleep. "Ding! Dong!" goes a bell. The friend wakes up and winks. "I'm Ged," he announces. "Who's out there?"

The child answers by typing in her name. The friend greets the child and asks if she'd like to play a game. If she would, the friend gives her a menu of the games in its repertoire.

If the child is using a disk-based system, the friend starts the game automatically. If the child is using a cassette-based system, the friend helps the child load the game program from tape.

After the game, the friend comes back on the TV screen. "I hope you had fun," it says. It offers to play a new game with the child.

The Friendly Operating System

The friend is like a simple operating system. It is the interface, the middleman, between your child and the computer. It is a first attempt at making computers warmer, more human and personable.

In coming months, we'll be gradually expanding the friend's capabilities. Next month, for example, I will write about a way tor the friend to learn more about the child. In a preliminary program the friend and the child will be "introduced." The child will give the friend personal information: name, age, the color of hair and eyes, address, phone number, likes and dislikes.

The friend will ask what kind of friend the child would like. The child will get a chance to mold the friend – to select the friend's name, shape, history, likes, and dislikes. If the child wishes, the friend can remain a computer. Or else the friend can become something completely imaginary and make-believe.

In fact, the child will be able to use the "Introduction" program to create several friends. The friends will have different characteristics and names.

If the child wishes, she can introduce the friends to each other.

Friendly Programming

The computer friends should liven up your child's computing. But they won't help with *programming*.

No matter how many games a friend has up its "sleeve," the child is never actually programming the computer. He or she is interacting with the friend and its programs. But not programming. Instead, in a way, the friend is programming the child.

This is one of the major drawbacks of the computer friends. They don't encourage children to write programs on their own. At least half of the value of the computer is unleashed when you program it yourself. Without that opportunity, your child is missing out on a lot.

Right now the friend is a friendly operating system. What we need is a friend that can also act as a *friendly computer language*. Then the friend can encourage the child to create, save, and run programs.

Beyond Logo

"Wait a second!" you say. "What about BASIC, PILOT, and Logo? These languages are easy to learn. They are friendly. They are perfect for kids."

Fred D'Ignazio is a computer enthusiast and author of several books on computers for young people. He is presently working on two major projects: he is writing a series of books on how to create graphics-and-sound adventure games. He is also working on a computer mystery-and-adventure series for young people.

As the father of two young children, Fred has become concerned with introducing the computer to children as a wonderful tool rather than as a forbidding electronic device. His column appears monthly in **COMPUTE!**.

My answer to that is: Do you have kids of your own? Do you teach kids? Have you ever tried to teach little kids how to use BASIC? Or PILOT? Or Logo?

I have two kids. My daughter Catie is almost seven. She's a first grader. My son Eric is three and a half. Eric spends his mornings at "Miss Eleven's Castle" (Evelyn's Day Care).

Both kids are whizzes at using the family computers. They have their own disks and tapes. They can turn the computers on and off, boot up disks, run programs, and key in the letters, numbers, and words the programs request. Both kids know all the special-function keys on the computer keyboard.

But try getting them to program? Forget it.

I can understand Eric's reluctance to program. After all, the kid doesn't even know how to read or write. If he gets 6's and 9's mixed up, and M's and W's, how can I expect him to master FOR/NEXT loops, string variables, subroutines, and arrays?

But Catie is a different matter. She reads Nancy Drew mysteries and "Choose Your Own Adventure" books. She is good at arithmetic, and she loves logic games, puzzles, and mazes. But she has no interest in programming.

Maybe it's just getting over the first hurdle. Unfortunately, Catie and I have been stuck on that hurdle for over two years.

The first hurdle is the first line of code in a program.

That first line is invariably a FOR/NEXT loop. The FOR/NEXT loop might do different things. It might print the message "CATIE LOVES MOWIE" a thousand times, all over the TV screen. (Mowie is Catie's kitty.) It might make the sound of a police siren or a dropping bomb, or the noise of water, or of crashing dishes. Or it might draw a drunken fly wandering across the screen.

What is Catie's reaction to all this? It's not positive, I'll tell you that.

Even if I get the fly to change into 16 different colors, Catie couldn't care less. After the first line of code, her reaction is sudden and dramatic. She gets hungry. Or she has to go to the bathroom. Or she has a headache. Or her spine dissolves and I get to watch her slide out of her seat and collapse into a puddle on the floor.

Or else she begins giggling and acts silly. She begins typing on the computer with her nose. Or her tongue.

This is an embarrassing situation.

On all sides we hear about friendly computers, computer literacy for kids, teaching kids to speak "computer" along with English. And here I am, a computer expert, a writer, an advocate for teaching computing to kids. So what do I do? I try to drag my kids into the computer age.

But they don't want to go.

Computer Literacy For Whom?

It's not so much that my kids resist me actively. It's just that they don't see the point. They have too much itching powder in their pants to make them sit still long enough to program.

At least using the languages available now.

But what if we created programming languages that incorporated the same ingredients as the best software designed for children? What are these ingredients? Quick response, for one. Other ingredients include: action, sound effects, pictures, colors. Quick mastery, a sense of power and control. Progress. Encouragement. Humor. These are qualities found in all good software for kids. But these qualities are not evident in programming languages. Even in PILOT. Even in Logo.

What Do You Think?

I hope I have lit some fires. Or started some fights.

What do you think? What kind of experiences have you had with your younger kids? Have they been similar to my experiences, or different?

Over the next few months as I continue to develop the computer friend and to write about other subjects, I plan to design and develop some prototype programming languages for little kids. The languages will be written in BASIC (or PILOT or Logo). They will be simple and experimental, something you can type into your computer and try yourself.

Also, the languages should contain the same qualities that make good programs so popular with kids. Maybe the programming will be in terms of colors, or sounds. Maybe in terms of shapes.

However it's done, the kids should be able to create programs themselves. They should be able to save, retrieve, and run those programs. The programs should not be trivial. They should do something. (Of course, they are doing *some*thing if they are teaching a child how to program.)

Most of all, the programming language should be fun for the kids to use. It should teach the kids that programming isn't something ugly that you have to do to get something nice. It's fun in itself. It's a way to express yourself, like coloring or playing music, or dancing.

The language shouldn't deter kids. It should encourage them to sit down and write a whole program. Even a short program.

Please write to me and tell me what you think. Send your letters to:

Fred D'Ignazio
c/o **COMPUTE!** *Magazine*
P.O. Box 5406
Greensboro, NC 27403

©

The Joy Of Joysticks

Atari/Commodore Game Controller Roundup

Tom R. Halfhill, Features Editor
Charles Brannon, Editorial Assistant

These joysticks and joystick substitutes will work with the Atari 400/800/1200XL computers, Commodore 64, VIC-20, and P Series computers, the Commodore Max Machine, the Atari VCS and 5200 video game machines, and the Sears Video Arcade. Some of the products, with proper adapters, also work with Apple, TRS-80, or Texas Instruments computers, and the Colecovision game machine.

Maybe you're playing a *Pac-Man*-type game....

Fleeing desperately from a relentless ghost, you make a break for the last energy pellet that will allow you to turn the tables on your pursuer. You try to round a corner in the maze, but suddenly find yourself slapping up against a wall. Why can't you turn? Blast that sticky joystick anyway! You've had it.

Or maybe you're playing a *Missile Command*-type game....

MIRVs and ICBMs are raining down and that infernal smart bomb is making straight for your last city. Quickly positioning the crosshair with the joystick, you take aim and fire your last ABM...and miss. If only you had a trackball like the one you're accustomed to in the arcades!

Or maybe you're playing an *Asteroids*-type game....

Hopelessly surrounded by an oncoming hailstorm of space debris, you yank back on the joystick to flip your spaceship into hyperspace, and find yourself dizzily spinning instead. Oh, for a hyperspace button like the one in the arcades!

The Joy Of Joysticks

Don't give up the spaceship – there is relief. A growing national obsession with home computer/video games has spawned an expanding market in custom game controllers. Only a year ago there were few alternatives to the common Atari-type joysticks supplied by the various manufacturers which use the Atari joystick standard. Now there are more than a dozen to choose from. The controllers covered in this overview were gathered after visiting computer stores, scanning magazine advertisements, and scouting new products at trade shows. While there are sure to be even more by the time this article appears, we tried not to leave any of the existing products out.

At first, it might seem that all joysticks must be more or less alike. Can there really be that much difference? After all, what is there to a joystick?

Externally, as the photos show, there is a wide range of configurations for joysticks (the name *joystick*, incidentally, originates from an early aviators' term for an airplane's control stick). Some joysticks are made to be hand-held and manipulated with a finger or two. Others are designed to rest on a tabletop and to be controlled with one hand. Some have hand-sized grips instead of short sticks. Some mount the fire button on the base, others on the stick, and still others have both.

Internally, there can be even greater differences. Some are constructed largely of plastic, others of metal. The construction largely accounts for a joystick's "feel." Since feel is a highly subjective reaction, we will avoid value judgments as much as possible. There is no substitute for trying a joystick yourself.

Some controllers, of course, are not joysticks at all. The push-button boxes are intended largely for *Asteroids*-type games, duplicating the arcade controls. Trackballs are at their best in games requiring rapid 360-degree movement, such as *Missile Command* and *Centipedes*.

And finally, a word about the standard Atari joystick. It's received some bad press, not all of it deserved. It's accused of being too fragile, unre-

...and so there were keys for the Atari 400.

In the beginning there was the membrane keyboard.

So it was to be done that Inhome Software would create a full-stroke keyboard for the Atari 400 Home Computer and it would be called the B Key 400, and would sell for $119.95 U.S. funds.

The new B Key 400 was made so easy to install that the owner could do it himself in a miraculous two minutes.

With the B Key 400 keyboard from Inhome Software, you will follow into the land of professional home computers that are powerful, easy to program and have a great capacity that can be made even greater with Inhome Software 48K and 32K memory boards. It was done and it was good.

INHOME

ADVANCING THE PROGRESS

Inhome Software Incorporated, 2485 Dunwin Drive, Mississauga, Ontario, L5L 1T1 (416) 828-0775. Atari is a trade mark of Atari Inc. Made in Canada sponsive, and even ugly. The joysticks do wear out after months of heavy use, but this isn't all the joystick's fault. First, in our experience, many "broken" joysticks are really the victims of faulty cords. The cords are subjected to a lot of twisting and pulling, and the thin wires tend to fray and snap. A dead joystick can often be revived by replacing the cord. Keep this in mind when admiring a custom joystick's hefty construction: the standard cord is probably its weakest link.

It might seem that all joysticks must be more or less alike. Can there really be that much difference?

Second, when an Atari joystick's joint or switches do break, it is often the fault of excessive flexing. Contrary to some beliefs, the Atari joystick *is* pretty responsive. Only a slight deflection is required to activate its switches. But its inherent stiffness, and the lack of any *tactile feedback* – that is, a positive click or snap when a switch makes contact – encourages people to wrench it harder than they have to. Games with slow joystick response, especially those written in BASIC, aggravate this problem.

Atari Joystick

Since the Atari joystick is the standard against which the others are most often compared, we'll start by pointing out that it's a two-handed instrument. Note that some joysticks permit one-handed operation, freeing the other hand for the keyboard (or for holding on to a chair).

Some people increase the leverage by jamming onto the end of the stick a PVC plastic "T" connector (available at hardware stores) or even a wine bottle cap. The Atari joystick includes a four-foot cord.

Atari Joystick
Atari, Inc.
1196 Borregas Avenue
Sunnyvale, CA 94086
$9.95 Each

Slik Stik

The Slik Stik is one of two joysticks by Suncom. Both resemble the Atari joystick, but incorporate some important differences. The Slik Stik's stick is only about half the height of the Atari's, but is topped by a jawbreaker-sized red ball for easy handling. And while the Slik Stik doesn't flex any more than the Atari stick, the action is more positive and you can feel a slight detent, or click. The fire button is very small but responsive.

The Slik Stik has a long six-foot cord reinforced at both ends with tough plastic collars where the cord joins the joystick base and plug. Suncom markets the Slik Stik as a direct replacement for the Atari joystick, and it is the only controller we reviewed which costs the same as the Atari product.

Slik Stik
Suncom, Inc.
270 Holbrook Drive
Wheeling, IL 60090
$9.95

Starfighter

Suncom's Starfighter, advertised as "The Ultimate Joystick," is very similar to the company's Slik Stik. However, Suncom claims it is more ruggedly constructed than their less expensive product, and it is guaranteed for two years instead of 90 days.

Where the Slik Stik has a ball-tipped controller, the Starfighter has a smooth plastic cylinder with a rounded top. It is taller than the Slik Stik, but still shorter than the Atari stick. The action is more positive, and the contacts in all eight positions can be distinctly felt. What's more, there are definite "stops" to the stick's movements, so it can't be damaged by over-twisting as the Atari joystick can. The Starfighter has the same convenient six-foot cord and reinforced connections as the Slik Stik.

Starfighter
Suncom, Inc.
$16.95

Baylis Big Stick

The Baylis Big Stick is the largest controller we tested. Actually, its name is something of a mis-

Introducing...the Byte Book Club

FORMERLY COMPUTER PROFESSIONALS' BOOK CLUB

POWERFUL TOOLS! POWERFUL SAVINGS!

Take any 3 books for $1.00 each only

(Values up to $68.50)

If you join now for a trial period and agree to purchase three more books—at handsome discounts—during your first year of membership. (Publishers' prices shown)

HOW TO BUILD YOUR OWN WORKING MICROCOMPUTER *By* C. Adams
582267-9 $16.95

AN INTRODUCTION TO VISICALC® MATRIXING FOR APPLE® AND IBM® *By* H. Anbarlian
016/054 $22.95

BUILD YOUR OWN Z-80 COMPUTER—*and*—Z80 USERS MANUAL. *By* S. Ciarcia & J. Carr
582337-3B $29.90
(Counts as 2 of your 3 books)

MINICOMPUTER AND MICROPROCESSOR INTERFACING *By* J. C. Cluley
582585-6B $27.50
(Counts as 2 of your 3 books)

BUSINESS INFORMATION PROCESSING WITH BASIC *By* G. Struble
582360-8 $17.95

BASIC: GETTING STARTED *By* W. S. Davis
582355-1 $5.95

INTERFACE PROJECTS FOR THE TRS-80 *By* R. C. Hallgren
582466-3 $18.95

PROJECTS IN MACHINE INTELLIGENCE FOR YOUR HOME COMPUTER *By* D. L. Heiserman
582574-0 $17.95

APPLE PASCAL GAMES *By* D. Hergert & J. T. Kalash
582521-X $19.95

TRS-80 GRAPHICS FOR THE MODEL I AND MODEL III *By* D. Kater & S. Thomas
333/033 $12.95

MICROCOMPUTER GRAPHICS AND PROGRAMMING TECHNIQUES *By* H. Katzan, Jr.
582576-7 $18.95

INTRODUCTION TO WORDSTAR™ *By* A. Naiman
582594-5 $21.95

TEACHING YOUR COMPUTER TO TALK: A Manual of Command and Response *By* E. R. Teja
582433-7 $15.95

INTRODUCTION TO COMPUTER ORGANIZATION *By* I. Tomek
582561-9 $23.95

PRINCIPLES OF INTERACTIVE COMPUTER GRAPHICS, 2/e *By* W. M. Newman & R. F. Sproull
463/387B $32.50
(Counts as 2 of your 3 books)

THE SMALL COMPUTER CONNECTION *By* N. L. Shapiro
564/124 $16.95

THE DEVIL'S DP DICTIONARY *By* S. Kelly-Bootle
340/226 $8.50

THE SOUL OF A NEW MACHINE *By* T. Kidder
582439-6 $13.95

APPLE PASCAL *By* P. Luehrmann
491/712 $16.95

MICROCOMPUTER OPERATING SYSTEMS *By* M. Dahmke
150/710 $15.95

WORD PROCESSING HANDBOOK *By* I. Flores
582645-3B $34.50
(Counts as 2 of your 3 books)

INVITATION TO FORTH *By* H. Katzan, Jr.
582284-9 $17.50

PROGRAMMING WITH ADA: An Introduction by Means of Graduated Examples *By* P. Wegner
789/24X $17.95

Why YOU should join the Byte Book Club now!

- **Best and newest books from ALL publishers!** Books are selected from a wide range of publishers by expert editors and consultants to give you continuing access to the best and latest books in your field.
- **Big savings!** Build your library and save money too! Savings range up to 30% or more off publishers' list prices—usually 20% to 25%.
- **Bonus books!** You will immediately begin to participate in our Bonus Book Plan that allows you savings of between 70%-80% off the publishers' prices of many professional and general interest books!
- **Convenience!** 14-16 times a year (about once every 3-4 weeks) you receive the Club Bulletin FREE. It fully describes the Main Selection and alternate selections. A dated Reply Card is included. If you want the Main Selection, you simply do nothing—it will be shipped automatically. If you want an alternate selection—or no book at all—you simply indicate it on the Reply Card and return it by the date specified. You will have at least 10 days to decide. If, because of late delivery of the Bulletin you receive a Main Selection you do not want, you may return it for credit at the Club's expense.

As a Club member you agree only to the purchase of three additional books during your first year of membership. Membership may be discontinued by either you or the Club at any time after you have purchased the three additional books. Orders from outside the U.S. cannot be accepted.

McGraw Hill

Fill out the card and mail today! If the card is missing, write to:

BYTE BOOK CLUB, P.O. Box 582, Hightstown, New Jersey 08520

nomer; the stick itself is only two and a half inches high, including the large red ball on the tip. It is the base that is big – nearly eight inches square. Obviously, the Baylis is designed to be rested on a tabletop or lap and operated with one hand.

The base is heavy enough to permit this kind of operation, although it does tend to rock around a bit during heavy action. However, there seems to be plenty of empty room inside the base to add weights, if you want to customize it. The stick itself is a rigid steel shaft built to tough arcade standards.

The response is very flexible and positive, with more "travel" than many joysticks. The fire button also is a large, arcade-style device. The cord is on the short side, only two and a half feet long, but since this oversized controller is not meant to be hand-held, this probably will not be a handicap.

The first Baylis Big Stick we sampled did not function in five of the eight directions. The internal switches were working perfectly, so the problem was traced to the cord. This is a perfect example of how even the most solidly constructed joystick can be paralyzed by the weakest link of any controller – its cord.

Baylis Big Stick
Released By:
Torrey Engberg Smith Co.
P.O. Box 1075
Glendale, CA 91209
$59.95

WICO Command Control

WICO's Command Control joystick is ruggedly built to arcade standards, with a steel shaft inside the plastic stick and metal parts at critical joints. This construction is not surprising, since WICO happens to be a major supplier of controllers for commercial arcade machines.

The Command Control joystick has a long "baseball-bat" handle, long enough to wrap your whole hand around. The action is smooth and flexible, with almost as much travel as the Big Stick. There is a small fire button on the tip of the stick and a larger one in the usual position on the base. A slide switch on the base selects between the two. The cord is five and a half feet long, strengthened with a plastic collar at the base end only.

WICO's product line includes two other joysticks, a trackball, extension cords, and adapters for Texas Instruments computers, the Radio Shack TRS-80 Color Computer, and the Apple II. The Red Ball joystick has a large ball mounted on the stick with the same base as the Command Control joystick, and the Command Control Deluxe features a batlike handle on a much larger base. All models are built to the same standards.

WICO Command Control
WICO-Consumer Division
6400 W. Gross Point Road
Niles, IL 60648
Command Control $29.95
Red Ball $34.95
Command Control Deluxe $39.95

Pointmaster

The Pointmaster is from Discwasher, a company whose best-known product is a popular cleaning system for phonograph records. The Pointmaster consists of a long plastic handle with a molded grip, attached with a ball joint to a plastic base. Since this unit is too light to use as a one-handed model, check to see if it is comfortable to use as a hand-held model, given its large size.

The stick is flexible enough, but there are no obvious contact points or "stops," so players should be careful not fo force the handle too far in the heat of video combat. Due to the stick's leverage and flexibility, precise positioning is sometimes difficult. The contoured fire button,

GAME PROGRAM DEVELOPMENT KIT

for the

COMMODORE VIC - 20

VIC - 20 is a registered trademark of Commodore Business Machines, Inc.

SIX TOOLS TO HELP YOU WRITE YOUR OWN FAST ACTION ARCADE-STYLE GAMES

DECODER — Decodes programs written in machine language (like game cartridges, utility cartridges, and even the computer's own internal operating programs). Produces a program in an English-like language (Assembler) which can be studied to figure out how they did it. The programs created with the decoder can be customized with the **EDITOR** AND INCORPORATED INTO YOUR OWN NEW GAME PROGRAM. The **ASSEMBLER** turns your programs created with the Decoder and the Editor back into machine language and puts them out to tape or disk so the **LOADER** can load them into the computer's memory to be tested and RUN. The **MONITOR** assists you in debugging your new game program by allowing you to run it a step at a time and making modifications if you need to. The **INSTRUCTION GUIDE** is written so that even a beginner can learn the skills needed to become a pro!!!

$49.95 plus $2.00 p&h buys the kit that could make you rich. Why wait?

Send check, M.O., VISA/MC ($2.00 s.c., please include expiration date), or specify COD (add $3.00) to:

French Silk smooth ware

P.O. Box 207, Cannon Falls, MN 55009

507-263-4821

mounted on the tip of the handle, has almost no travel. When first toying with the Pointmaster, without plugging it in, we feared the button would have a "dead" feel. But actually, it turned out to be very sensitive and fast.

The cord is five feet long, reinforced with a collar at the base end only.

Pointmaster
Discwasher, Inc.
1407 N. Providence Road
Columbia, MO 65201
$16.95

Quick Shot

Spectravision's Quick Shot joystick has one unique feature that interested us immediately – the four rubber pads that are standard on other joysticks can be removed and replaced with four suction cups. This allows Spectravision to make the joystick small and light enough to be hand-held, yet still capable of being anchored firmly to a tabletop for one-handed use without resorting to a huge base or extra weights. We found, however, that the tabletop must be very smooth for the suction cups to stick, even if they are moistened.

Plastic construction dominates in the Quick Shot. The stick is a large, molded pistol grip that fits an adult's hand better than most of the other joysticks we tested. The action is flexible, with definite stops, although the contact points are hard to feel. There are two fire buttons, one on the stick and another on the base, and both are always "live," so you can switch back and forth in mid-action. The buttons also have a detent, or "click," at the bottom of their travel.

The Quick Shot includes a four-foot cord strengthened at the base end only.

Quick Shot
Spectravision
39 W. 37th Street
New York, NY 10018
$14.88

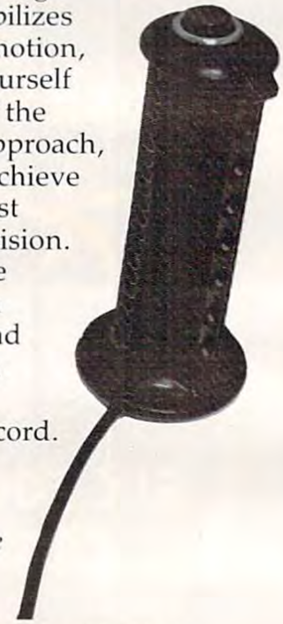
Le Stick

Le Stick is the most unusual joystick we tested. Datasoft claimed in early magazine ads that Le Stick was adapted from Air Force designs for advanced controllers. Le Stick consists only of a joystick – no base. Constructed of a pliant, rubber-like plastic, the handle incorporates four mercury switches which are activated by tilting. That is, tilting the handle forward causes the screen object to move up, tilting it backward moves the object down, and so forth.

This ingenious approach seems to have several advantages: without a mechanical connection to a base, flexibility is unlimited; there is no ball joint to wear out; true one-handed operation is possible, since the fire button is tip-mounted; and the joystick is very light.

However, since the joystick has no "self-centering" or definite "up" position relative to an attached base, it can be difficult to maneuver for those accustomed to conventional joysticks. For example, our untrained hands found it difficult to tilt horizontally without mixing in some vertical motion, and vice versa. Although squeezing the handle immobilizes the sensor and cancels any motion, it can be hard to re-orient yourself without taking your eyes off the screen. As with any novel approach, practice will be required to achieve mastery – we suggest you test Le Stick before making a decision. Our last suggestion – beware the "grip of death" when, in panic, your hand clinches and immobilizes the joystick ... a calming challenge.

Le Stick has a four-foot cord.

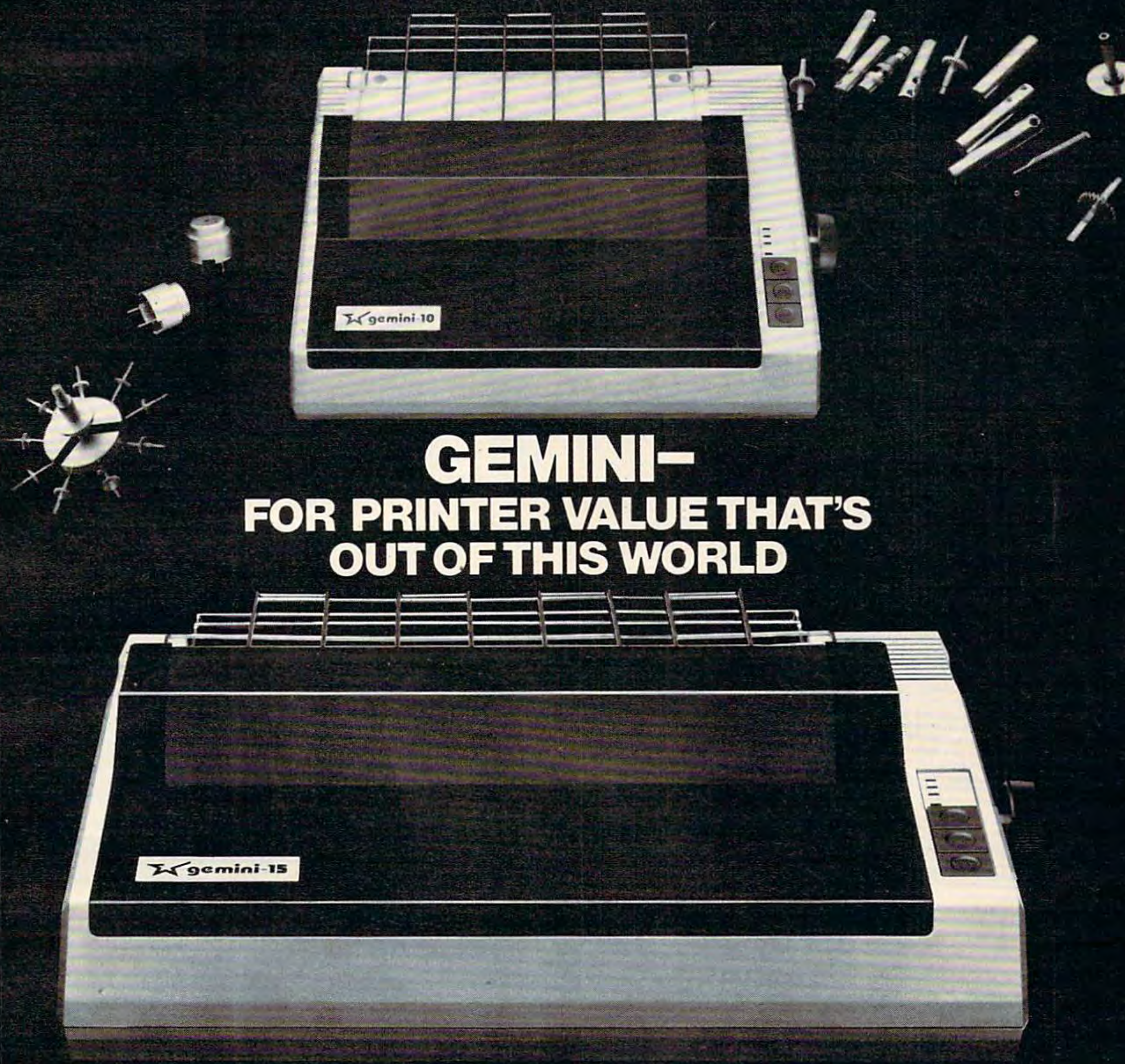


Le Stick
Datasoft, Inc.
19519 Business Center Drive
Northridge, CA 91324
$39.95

Starplex Video Game Controller

Unlike the joysticks reviewed, the Starplex Controller and the KY Enterprises box covered below are not really general-purpose devices suitable for all types of computer games. Instead, the Controller is intended largely for one game – *Asteroids*. The button layout is designed to simulate the controls on the commercial arcade version. Thus, we find buttons labeled "Left," "Right," "Up," "Down" (Hyperspace), and "Fire." These correspond to the rotational, rocket, and panic buttons on the arcade machine.

The Starplex Controller fulfills its task very well. Anyone accustomed to playing *Asteroids* in the arcades will feel much more at home with these large, sensitive buttons than with a joystick. One interesting feature is the "Astroblast." Selecting this option with a slide switch allows automatic repeat when the fire button is held down. In other



GEMINI–

FOR PRINTER VALUE THAT'S OUT OF THIS WORLD

Over thirty years of down-to-earth experience as a precision parts manufacturer has enabled Star to produce the Gemini series of dot matrix printers—a stellar combination of printer quality, flexibility, and reliability. And for a list price of nearly 25% less than the best selling competitor.

The Gemini 10 has a 10″ carriage and the Gemini 15 a 15½″ carriage. Plus, the Gemini 15 has the added capability of a bottom paper feed. In both models, Gemini quality means a print speed of 100 cps, high-resolution bit image and block graphics, and extra fast forms feed.

Gemini's flexibility is embodied in its diverse specialized printing capabilities such as super/sub script, underlining, backspacing, double strike mode and emphasized print mode. Another extraordinary standard feature is a 2.3K buffer. An additional 4K is optional. That's twice the memory of leading, comparable printers. And Gemini is compatible with most software packages that support the leading printers.

Gemini reliability is more than just a promise. It's as concrete as a 180 day warranty (90 days for ribbon and print head), a mean time between failure rate of 5 million lines, a print head life of over 100 million characters, and a 100% duty cycle that allows the Gemini to print continuously. Plus, prompt, nationwide service is readily available.

So if you're looking for an incredibly high-quality, low-cost printer that's out of this world, look to the manufacturer with its feet on the ground—Star and the Gemini 10, Gemini 15 dot matrix printers.

star®
MICRONICS • INC
MAKING A NAME FOR OURSELVES

1120 Empire Central Place, Suite 216, Dallas, TX 75247
For more information, please call Bob Hazzard, Vice President, at (214) 631-8560.

words, now you can machine-gun the nasty asteroids. This feature requires an AA battery to be installed inside the controller.

The Starplex also works well with *Space Invaders* and other games requiring simple up-down or left-right movement. For games that demand complex 360-degree movement, stick with a joystick. Obviously, you'll have to decide if you can use this type of controller often enough to justify its cost.

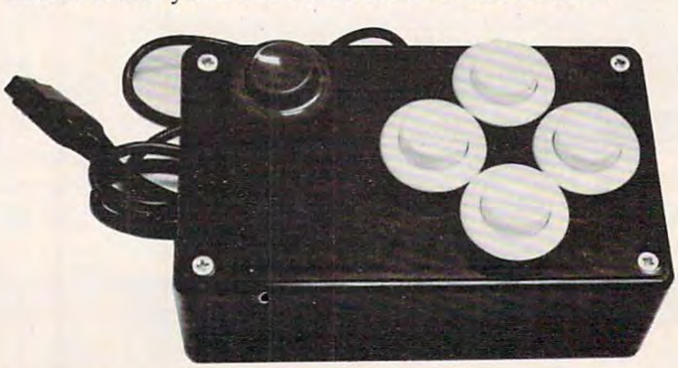
The Starplex is light enough to rest on a lap, and stable enough to hold still on a tabletop. It has a four-foot cord reinforced at the base end only.

Starplex Video Game Controller
Starplex Electronics, Inc.
E23301
Liberty Lake, WA 99019
$29.95

Fingertip Controller

This controller is very well constructed, with a heavy metal box and five large, springy, arcade-style buttons. The buttons are unlabeled, but the white ones correspond to up, down, left, and right, while the red one is the fire button.

Although you can achieve diagonal movement by simultaneously pressing both a vertical and horizontal button, the Fingertip Controller seems most suited to games with simple up-down or left-right movement, such as *Space Invaders*. Like the Starplex Controller, it also works well for *Asteroids*, but with a quirk – it's left-handed. That is, your right hand controls the rotational movement while your left hand hits the fire button, just the opposite of the arcades.



As per the instructions, it's easy to adjust the sensitivity of the buttons by opening the box and bending the spring switches. The Fingertip Controller has a five and a half-foot cord.

Fingertip Controller
KY Enterprises
3039 East Second Street
Long Beach, CA 90803
$26.95

Command Control Trackball

True arcade fans have been hungering for one of these for a couple of years now. Commercial arcade games which use trackballs – such as *Missile Command* and *Centipedes* – work okay when trans-

lated to joysticks in home versions, but the "feel" just isn't there. And since the avid arcade fan strives to re-create the arcade experience as closely as possible, joysticks sometimes just don't quite measure up.

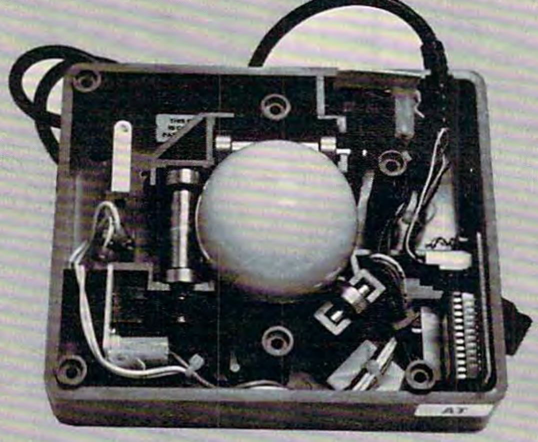
Since WICO supplies trackballs for commercial arcade machines, you would expect the company's home version to be similarly well-constructed – and you won't be disappointed. The heavy billiard-style ball rotates quite smoothly and "coasts" with a good spin. This is due to high-quality steel shafts with ball bearings (see the accompanying sidebar and inside photo describing how the trackball works). Even the five-foot cord is extra heavy-duty. The trackball's inherent weight and rubber footpads keep it from sliding around on a tabletop, and the fire button is the same as those found on WICO's joysticks.

As an example of what a trackball can do in a game demanding fast 360-degree movement, one of our testers tried it out on Atari's *Missile Command*. His former high score was 39,000. With the trackball, after a few warm-up games, he scored 66,000.

Command Control Trackball
WICO
$69.95

WICO Trackball: The Inside Story

Ottis Cowper, Technical Editor



Most joysticks operate by opening or closing switches as the handle is moved. In the standard Atari configuration, four switches provide a four-bit binary number for control of motion in eight distinct directions.

Exceptions to this rule are joysticks such as those used with the TRS-80 Color Computer which use a pair of potentiometers (variable resistors) to provide varying voltages which must be converted by the computer to meaningful binary values. Such joysticks are essentially two-dimensional game paddles. The WICO trackball uses an altogether different technique. Let's take a look inside this rather unconventional game controller to see how it works.

The ball, which is remarkably similar to a billiards cue ball, rests on three rollers with ball bearings for smooth motion. The two larger rollers, one placed vertically and one horizontally, both have a shaft with a slotted disk on one end. These disks pass through the gap in an electronic device known as a *photon-coupled interruptor* and herein lies the key to the trackball's operation. A photo interruptor consists of a light-emitting diode (LED) and a phototransistor separated by a gap. As long as the gap is not obstructed, light from the LED strikes the phototransistor and turns it on. If the light is blocked, the transistor turns off.

As the slotted disk rotates, an alternating series of solid sections and holes passes through the gap, causing the transistor to toggle on and off as light from the LED is alternately blocked and allowed to pass. (The photo interruptors make it possible to determine in which direction the disk is rotating.) Since the transistor can be thought of as an electronic switch, this has the same effect as pushing the joystick handle in one direction, except that the input is much faster and smoother.

For games which require rapid motion all over the screen, the trackball is a major improvement, although the standard joystick is probably more suitable for applications which require precise positioning.

TG Trackball

This trackball should be on the market by the time you're reading this issue. The unit we tested was a prototype that we obtained at the COMDEX trade show in Las Vegas. TG Products also is introducing an Atari plug-compatible joystick, but we were unable to obtain one of these for testing.

The TG Trackball works much like the WICO Trackball, using LEDs and phototransistors to detect the ball's spin. The plastic ball glides less smoothly than the WICO's, however, and has much less tendency to coast. Approximately one third of our testers preferred this "feel" for fine positioning, so this is purely a personal matter that should be tested by the purchaser. Inside, the TG Trackball supports the "billiard ball" on plastic shafts without ball bearings. It might be a good idea to lubricate these shafts to reduce excessive wear if this hasn't been done in production models.

The trackball's extra-heavy cord is just short of five feet and is reinforced at both ends.

TG Trackball
TG Products
1104 Summit Avenue
Suite 110
Plano, TX 75074
$64.95

©

HOFACKER

Books + Software for ATARI VIC-20 OSI SINCLAIR TIMEX

Payment: check, money order, VISA, MASTER-CHARGE, Euroscheck.
Orders from outside USA: add 15% shipping. CA residents add 6.5% tax
*ATARI is a registered trademark of ATARI Inc.
*VIC-20 is a registered trademark of Commodore

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

PRODUCTS FOR ATARI* 400/800 FROM ELCOMP

BOOKS:

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**

Programming 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 #166 **$19.95**

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

SOFTWARE IN BASIC FOR ATARI

Invoice Writing for Small Business
This program makes writing invoices easy. Store your products in DATA statements with order-number, description, and price. The program later retrives the description and price matching to the entered order-number. The shipping cost and the discount may be calculated automatically depending on the quantity ordered or entered manually. The description to the program tells you how to change the program and adapt it to your own needs. Comes with a couple of invoice forms to write your first invoices on to it.
Order #7201 cassette version **$29.95**
Order #7200 disk version **$39.95**

Mailing List
This menu driven program allows the small business man to keep track of vendors and customers. You can search for a name or address of a certain town or for an address with a certain note. 50 addresses are put into one file.
Order #7212 cassette version **$19.95**
Order #7213 disk version **$24.95**

Inventory Control
This program is menu driven. It gives you the following options: read/store data, define items, entry editing, inventory maintenance (incoming-outgoing), reports. The products are stored with inventory number, manufacturer, reorder level, present level, code number, description.
Order #7214 cassette version **$19.95**
Order #7215 disk version **$24.95**

Programs from Book #164
The programs from book no. 164 on cassette. (Book included)
Order #7100 **$29.00**

Game Package
Games on cassette. (Bomber, tennis, smart, cannon fodder, etc.)
Order #7216 **$9.95**

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

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**

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, Hex-dump (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 Mysferies
A complete guide to customizing the Apple Software und 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!

COMING SOON! ORDER NOW!
A Look in the future with your ATARI
(Astrology and how to do your own horoscope on the ATARI 800. **Order No. 171** **$9.95**
FORTH on the ATARI – Learning by Using
Order No. 170 **$7.95**

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 **$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 Biorythm for ATARI (cass. or disk).
Order #7223 $29.95
Birth control with the ATARI (Knaus Ogino)
Order #7222 cass. or disk $29.95

Books + Software for VIC-20 (requires 3k RAM Exp.)
#4870 Wordprocessor for VIC-20, 8k RAM $19.95
#4883 Mailing List for VIC-20, 16k RAM $14.95
#141 Tricks for VICs - The VICstory Progr. $9.95
#4880 TIC TAC VIC $9.95
#4881 GAMEPACK I (3 Games) $14.95
#4885 Dual Joystick Instruction $9.95

INPUT/OUTPUT Programming with your VIC
Order #4886 $9.95
#4896 Miniassembler for VIC-20 $19.95
#4881 Tennis, Squash, Break $9.95
#4894 Runfill for VIC $9.95

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. The board contains a large prototyping area for your own circuit design and expansion. The construction article shows you how to build your own 3k RAM expander and ROM-board.
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 #140 (book) $9.95

Books for OSI
#157 The First Book of Ohio $7.95
#158 The Second Book of Ohio $7.95
#159 The Third Book of Ohio $7.95
#160 The Fourth Book of Ohio $7.95
#161 The Fifth 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**

C. Regena

Write Your Own Games

Some tips on getting the most out of your TI when writing games.

You have probably discovered that one of the fun things to do with your TI-99/4A is to play games. In fact, many people who wanted one of the popular game machines have discovered that for about the same amount of money they could have a *computer* and still be able to play games. Many of the games written for the TI-99/4A are arcade quality – that is, they have good graphics and fast action.

The programs on the command modules can be programmed in UCSD Pascal, TMS9900 Assembly, and Graphics Programming Language (GPL). These languages take maximum advantage of the color, graphics, sound, and speech capabilities of the computer. GPL is an excellent language for drawing graphics and allows the speed of an assembly language.

To program your own games with fast, smoothly-moving objects, you will want to use TI Extended BASIC. It allows you to use up to 28 "sprites." You may define the shapes of the sprites and designate a certain magnification. You may also specify the sprites' speed. The row velocity and the column velocity may vary from -127 to +127, and by specifying numbers for both velocities you will get a diagonal movement. Sprites "wrap" at the edges of the screen, so you don't need to worry about "crashing" your program on edge conditions. With *one* CALL SPRITE statement you can define the sprite number, shape, color, position, and speed.

TI Console BASIC (the BASIC built in with no accessories or peripherals) is a language powerful enough that you can design a variety of fun games with it. If you have moving objects, however, they have to move a square at a time and thus will have jerky movement. Depending on the number of objects, BASIC games tend to be slow; however, I have seen several fast action games that really require nimble fingers.

Whether you are writing a game in TI BASIC or in TI Extended BASIC, I can offer a few programming tips. Keep in mind that the best way to learn is to actually start programming – and playing.

Randomness

Probably a central tool in computer games is the machine's ability to choose things randomly. Most computers have the command RND, but each computer has a slightly different syntax (way of writing the command). On the TI-99/4A, RND represents a random number between zero and one. Turn on your computer, press any key to begin, and press 1 for TI BASIC. Now type in PRINT RND and press ENTER. The computer will print a decimal fraction (to ten places). Usually in game situations you won't want a fraction, so multiply that fraction by a number. For example, multiply RND by 10 like this: PRINT 10*RND or PRINT RND*10. Now you will get ten times that decimal fraction.

You probably want just the whole number part of that mixed decimal number. Use the INTeger function to get the whole number. PRINT INT(10*RND). If you keep trying this command, you will get numbers from zero to nine. Remember, INT truncates the decimal portion; it does not round the number. Suppose you really wanted random numbers from one through ten. The command would be: PRINT INT(10*RND)+1 or PRINT INT(10*RND+1).

One more step. Assume you want a number N to be a random number between 10 and 20, inclusive. 20 - 10 = 10. There are 10 numbers plus 1 ("inclusive"). The command could be N=INT(11*RND)+10. The portion INT(11*RND) will give you numbers from 0 to 10; then you add 10 to get numbers from 10 to 20.

Now try this short program:

```
100 FOR I=1 TO 10
110 PRINT INT(10*RND)+1
120 NEXT I
```

RUN the program. RUN it again. And again. The program is printing ten random numbers from 1 to 10. However, you'll notice that each time you run it, you get the same numbers in the same order. You need to add the line: 105 RANDOMIZE.

The RANDOMIZE command mixes up the numbers so that each time the program is run you will get different numbers – and that's what you want in a game. The *User's Reference Guide* indicates that the RANDOMIZE statement only needs to be somewhere in the program to generate different numbers; however, I have found that one RANDOMIZE statement at the beginning of a program does not always work. It is better to use the RANDOMIZE statement just before you use the statement containing RND. Note: If you are debugging a program, you may want to leave RANDOMIZE out so you'll know exactly what numbers your program is choosing. Debug your program, then add the statement and test it.

Moving Objects

In general, the fewer moving objects you have in your game, the faster the action can be, and the logic will be a lot less complex. Also, each moving object should be specified by only one character number so you don't have to use up valuable time by building an object out of several characters. To move an object in TI BASIC you need to erase the object in the first position (replace it with a space) and draw it again in the second position – each move takes two statements.

Player Input

There are two main ways the computer can understand what you want: by your using the joysticks or pressing keys on the keyboard. Your game may be designated for joysticks only, keyboard only, or both. Because of the logic involved, a game using both methods of input will be slightly slower in response; and depending on the branching sequence, one of the methods will be slower than the other.

Joysticks may be easier to use to learn a game, especially if the player is used to a video game using joysticks. My own children, and many other players I know, prefer using the keyboard for *TI Invaders* and *Munchman* because the joystick response is considerably slower than the keyboard response.

The keyboard action is easy to learn because there are standard arrow keys for all games designed for the TI-99/4A. Programmers writing games for other computers often choose their own favorite keys to use, and the directions are different for each game. On the TI-99/4A, the arrow keys are E (up), X (down), S (left), and D (right), with the shooting key either the ENTER key or the period key. If there are two players, the standard arrow keys on the right half of the keyboard are I, J, K, and M. The TI-99/4 owners have a slight advantage here – there is an overlay available for the old keyboard that shows the arrow keys, and it is easier to use the old keyboard for two-player games.

The TI joysticks (wired remote controllers) come with a little instruction book with some sample programs. The main command is CALL JOYST(K,X,Y), which returns an X and Y value for the position of the joystick, where X and Y may be 4, -4, or 0.

To detect keys pressed on the keyboard, use the CALL KEY command. This command is like the GET command in other BASIC languages. The form is CALL KEY(0,KEY,STATUS) where 0 means to scan the whole keyboard. STATUS is a variable name (it could be ST or S, or whatever you wish) which will return whether a key has been pressed or not. KEY is a variable name (again, use whatever you wish) that will return the ASCII code of the key pressed, such as 13 for the ENTER key, 65 for the letter A, 69 for the letter E, etc.

By using IF statements, you can check which key was pressed and branch accordingly. You can also GOTO the CALL KEY statement for other keys to make the computer act as if it is ignoring all responses except the keys allowed. Here is a sample using arrow keys:

```
100 CALL KEY(0,K,S)
110 IF K=69  THEN 1000      (up arrow)
120 IF K=68  THEN 2000      (right arrow)
130 IF K=88  THEN 3000      (down arrow)
140 IF K=183 THEN 4000      (left arrow)
         ELSE 100           (any other key will be ignored)
```

Remember, there are several ways to program the same procedure; this is just one way. You may prefer to use "not equal" signs or a split keyboard and an ON GOTO statement.

A split keyboard approach scans half the keyboard using CALL KEY(1,K1,S1) or CALL KEY(2,K2,S2). The key codes returned for up, right, down, and left are 5, 3, 0, and 2. A sample program using the split keyboard is:

```
100 CALL KEY(1,K,S)
110 IF (K<0)+(K>5)THEN 100
120 ON K+1 GOTO 3000,100,4000,2000,100,1000
```

Line 110 makes sure the K value is in the right range; the key value must be from 0 to 5. All other keys are ignored. Line 120 branches according to which key was pressed. The keys corresponding to 1 and 4 were not acceptable, so they return to the CALL KEY statement. If you want to try out either of these programs, add the following lines, then RUN and try pressing various keys.

```
1000 PRINT "UP"
1010 GOTO 100
2000 PRINT "RIGHT"
2010 GOTO 100
3000 PRINT "DOWN"
3010 GOTO 100
4000 PRINT "LEFT"
4010 GOTO 100
```

There is a slight problem in testing for zero on the TI-99/4A console. Use logic such as IF K+1<>1 rather than IF K<>0. Also, some of the split keyboard codes are different for the TI-99/4A than for the TI-99/4. It is better not to use the comma, period, semicolon, slash, space bar, ENTER, SHIFT, B, and G so that programs may be used on either console. ©

Five VIC Games From Nufekop

David Malmberg

This latest batch of Nufekop games once again proves the company is worthy of its name. The word *Nufekop*, according to the firm's early ads, has a Druid origin, and means putting an extraordinarily large amount into a small pocket or enclosure, possibly through the use of magic. This is an apt name for a software company that can pack so much fun, excitement, fantastic sound, and colorful graphics into its programs and get them to fit into the VIC-20's relatively small memory.

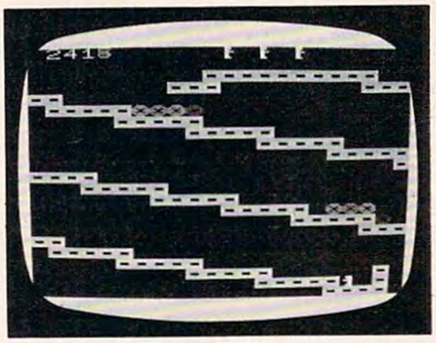
Before describing the individual games, let me explain the evaluation criteria. I believe the most important attribute of a great game is its "lasting power." It should be just as much fun to play the game the hundredth time as the first or second time. You shouldn't become bored or jaded. Ideally, the game should have multiple levels of difficulty. The game shouldn't be too easy for the expert or too hard for a beginning player. A great game will make you want to play it again and again – or as they say in the coin-operated video game trade, a great game is one which will keep you "pumping in the quarters."

In evaluating these games, I made use of a panel of expert consultants – the neighborhood children from 8 to 14 years old. Each was asked to comment on the things he or she liked and disliked and to rate each game on a scale from zero to ten. A zero rating means it is a waste of time to play the game even *once*. A ten means it's as good as the best full-fledged arcade games, for example, *Centipede* or *Pac-Man*. The comments and ratings that follow reflect the consensus of these experts, as well as my own opinions.

Krazy Kong

The object in this game is to rescue the maidens from the evil Kong's clutches while he tries to stop you by hurling barrels down at you. There are various configurations of steps to climb. You may use either the keyboard or a joystick to climb the steps and jump over the barrels. As you save each maiden, you are presented with a new set of steps – each harder than the last. The game ends when you are killed by a falling barrel, run out of energy, or have rescued all three maidens.

Krazy Kong is well done. It has great graphics, sound effects, and music. The action is very fast. There are four levels of play that govern the number of barrels and the length of your jumps. The highest level is tough enough to challenge even the most seasoned gamester.

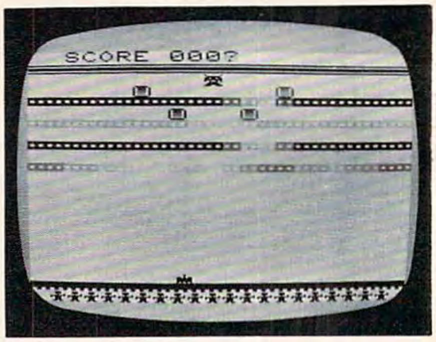


Barrels tumble down the stairways as the little man begins his ascent to rescue the maiden in Krazy Kong .

However, my experts were a bit disappointed that *Krazy Kong* didn't have a little more variety in the paths up to the maidens and in the obstacles to dodge. Challenging though it was, they quickly became bored with climbing steps and jumping barrels. *Krazy Kong* doesn't have the lasting power of a really great game, so the consensus rating was seven out of a possible ten. *Krazy Kong* works in a standard 5K VIC and is priced at $12.95.

Anti-Matter Splatter

This game is difficult to describe. Anti-matter "bombs" are falling to earth. You control a splatter-matter cannon using either the keyboard or the joystick. You try to shoot the bombs with your cannon before they hit the people at the bottom of the screen and make them disappear. (What else would you expect an anti-matter bomb to do to a person?) As the game progresses, the action gets increasingly frantic



Deadly anti-matter bombs drop from the sky toward your people in Anti-Matter Splatter.

WIN $5,000 Plus Royalties!*

For the best Talking Game

For the Atari® or Apple® II+

Using the VOICE BOX™ Speech Synthesizer

Now you can make your Atari® 400/800 or Apple® II games and other programs come alive with the VOICE BOX by the Alien Group — the first low-cost, smart speech synthesizer with unlimited vocabulary.

Add jokes to your programs. Insults. Compliments. Help messages. Stories. Alien voices. Animal roars. Have your computer talk to the fire department or police in emergencies. To kids. Or blind people. Teach touch typing with immediate spoken feedback. Or just about any other subject — the fun way. Or help a speech-impaired friend communicate . . . the possibilities are limitless.

The VOICE BOX plugs into your Atari's serial port. And talks directly through your TV set. Or into any Apple II slot. No power supply or special interfaces needed.

Just select from its simple screen menu. A dictionary with thousands of common words (on diskette or cassette) automatically translates your text into speech. It's that easy.

But don't let its friendliness fool you. The VOICE BOX has all 64 phonemes (basic sounds, like "ah") built in. So you can precisely create any word or sound you can imagine. And store it all on diskette or tape. Names or foreign language words, for example. Or wierd non-human languages.

Let me entertain you — The VOICE BOX is creative too. It will crack you and your friends up with non-stop random, grammatically correct sentences, using words you specify. It also has an amusing talking face with lip-sync animation — a real crowd-stopper. Best of all, you can call the VOICE BOX from any BASIC program and make your program really hum —literally!

Singing Apples? — Apple owners get all these capabilities too — as a plug-in card plus diskette. Or there's a deluxe version with the dictionary in ROM (no diskettes to bother with), speaker, and ability to "sing" (hey we're not making this up folks) in any key. (Both Apple versions require 32K or more. Applesoft and DOS 3.3).

Don't confuse the VOICE BOX with "dumb" speech synthesizers that can't learn new words. Or software-based ones with lower speech quality — and an annoying tendency to blank out the display when they talk. The VOICE BOX is a true breakthrough in speech synthesis. Small wonder thousands of Atari and Apple owners have already bought the VOICE BOX.

The VOICE BOX is available now at leading computer stores throughout the world. Or direct from the Alien Group, with 10-day money back guarantee if you're not completely satisfied.

VOICE BOX For Atari. $169.00
16K and 32K versions included
(Specify diskette or cassette).

VOICE BOX for Apple II.+ $139.00.
(Requires speaker.)

VOICE BOX for Apple II+. $215.00
(Includes dictionary in ROM and singing capability. Comes with speaker.)

Enclose check or money order.

CHECK YOUR LOCAL COMPUTER STORE FOR TALKING "VOICE BOX" VERSIONS OF YOUR FAVORITE GAMES FROM LEADING GAME COMPANIES

*Win a $5,000 prize — plus royalties — for the best Atari 400/800 or Apple II+ game using the VOICE BOX. Deadline: May 30, 1983. Write for contest details.

Please mail to: The Alien Group, Department PT-6, 27 W. 23 St., N.Y., N.Y. 10010
Or call in order to (212) 741-1770

with more bombs and greater speed.

You lose the game whenever a bomb hits your cannon or whenever all of the people have been reduced to anti-matter. You score points by shooting down the bombs, but the high score is not saved. No one who tried this game was ever able to "win," so it is not clear how (or if) it is possible.

Anti-Matter Splatter is written entirely in machine language so the speed is incredibly fast. The graphics and sound are outstanding. This game has good lasting power; the kids played it again and again.

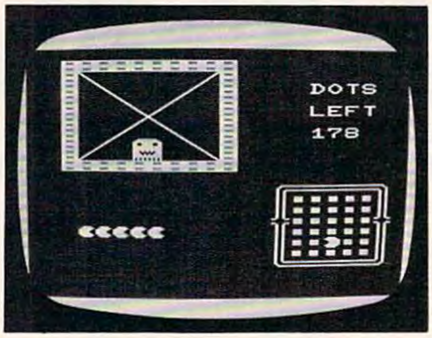
However, the game could have been improved. You get only one cannon, so the game is often over before it has barely begun. The high score should be displayed, so players would have something to try to beat. A variable level of difficulty would be a nice improvement. This could be done by varying either the speed and/or the number of cannons.

Anti-Matter Splatter was rated an eight out of ten. The program works in a standard 5K VIC and retails for $24.95.

3-D Man

3-D Man is a very clever idea for a game – you move through a maze that is displayed in three dimensions. Long corridors with occasional passageways on the sides are displayed in perspective. The object of the game is for your 3-D Man to eat all of the dots, before he is eaten by one of the four ghosts that randomly roam the maze. During the game, the screen shows what your 3-D Man sees ahead of him. At the same time, a small radar screen shows the overall maze and your 3-D Man's location and direction within it. The score corresponds to the number of dots gobbled. You get five 3-D Men before the game is over.

The graphics of *3-D Man* are extremely fast and superbly



In 3-D Man, *players must eat dots in a three-dimensional maze (upper left) while watching out for pursuers on the bird's-eye view map (lower right).*

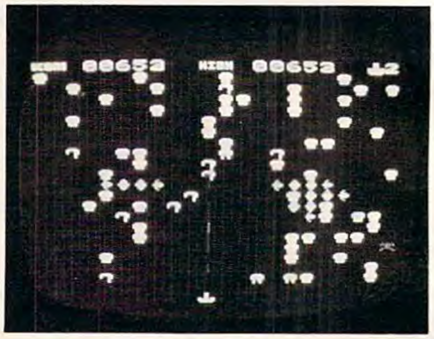
done. Sound is very effectively used, with different noises for such events as eating a dot, being eaten by a ghost, or trying to make an illegal movement (i.e., bumping into the maze wall).

3-D Man, however, is an example of a game that lacks lasting power. Everyone loved it the first couple of times. As they continued to play, they discovered its major flaw – that the ghosts' positions are truly random; they do not move from one location to a contiguous one. As an example, it is quite common to encounter a certain ghost (e.g., the red one), then turn around and attempt to flee in the opposite direction, only to find the same ghost there, too. Because of the random nature of the ghosts, *3-D Man* is not really a game of skill. With success so dependent on luck, all of my experts soon lost their enthusiasm. The consensus rating was a five.

3-D Man requires a 3K memory expander and a joystick. It is priced at $19.95.

Exterminator

This is one of the best games I've ever seen for the VIC or any other computer. The object is to shoot everything that moves and everything that doesn't. You normally have three shooters, but you can get a free one at 5000 points. Spiders speed up when you get to 20,000 points. The screen changes color combinations whenever you annihilate all of the pieces of the current centipede.



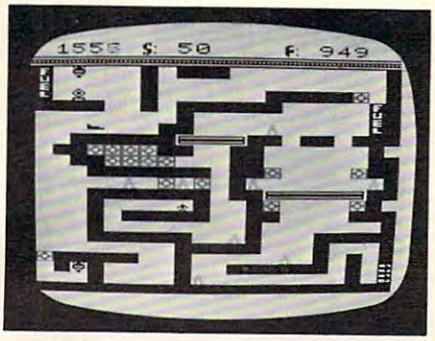
Blasting away at centipede sections, spiders, mushrooms, and other obstacles in Exterminator.

Exterminator is an absolute marvel! Written entirely in machine language, it is unbelievably fast. The graphics, sound, and music are all fantastic. This game is clearly the most popular in my library. The fact that Nufekop was able to fit all of this action and fun into a standard 5K VIC without any additional memory is a tremendous accomplishment. The rating was unanimous among my panel of experts – ten out of ten. *Exterminator* is a great buy at $24.95.

Defender On Tri

The object of this game is to save a group of scientists who have become trapped while exploring an abandoned space station (with the code name "Tri") before the station crashes into the sun. Using the joystick, you control a small rescue vessel. Unfortunately, your ship has room for only one passenger – so you must find the scientists and bring them safely through the maze of machinery in the space station one at a time. This is a very hazardous journey, since the machinery is moving very fast and will destroy your ship unless your defense shields are activated.

However, you cannot have your shields energized too often because they drain so much of



In Defender On Tri, *players must maneuver a tiny ship (upper left) through a maze of machinery in a huge space station to rescue scientists.*

your ship's fuel that you would be unable to complete your mission. You are in a dangerous race against the clock. Time is running out. As Tri moves closer and closer to a collision with the sun, the machinery begins to speed up. You have precious little fuel left and have to make every drop count as you thread your way through a maze where one false move means sudden death.

This game, too, is exceptionally well done. The graphics are great. The action is fast. The sound effects are good. The game is quite exciting, although it is very difficult. The only complaint anyone had was that the game was probably too difficult. None of the neighborhood kids was ever able to rescue all of the scientists. Several kids got frustrated and gave up on the game. Still, the consensus rating was a high nine out of ten.

Defender On Tri requires a 3K memory expander and a joystick. It retails for $19.95.

Nufekop games are widely distributed. The games may also be purchased directly from the company.

Krazy Kong
Anti-Matter Splatter
3-D Man
Exterminator
Defender On Tri
Nufekop Software
P.O. Box 158
Shady Cove, OR 97539
5K to 8K RAM
$12.95 to $24.95

©

Apple Game Animation Package

Michael P. Antonovich

The *Game Animation Package* is marketed by Synergistic Software as a two-part program package: *Fast Draw* and *Micro Sketcher*. *Micro Sketcher* is for creating high-resolution color pictures which can be used as backgrounds in games or other programs. *Fast Draw* is an excellent graphics utility for creating bit-mapped shape tables which can be accessed from BASIC programs to achieve fast, smooth, flicker-free action.

Fast Draw

First, let's look at *Fast Draw*. Have you ever wanted to be able to create shape tables in color? Or to move shapes around the screen at lightning speed without screen flickering as you draw and erase the shapes? I, for one, was glad to see this package. I found *Fast Draw* a very easy way to create and manipulate shapes.

Fast Draw consists of four major program segments which allow the user to create and manipulate bit-mapped shape tables. The four segments are:

Delimit – This program is a shape table editor which allows you to create shapes using any combination of the eight standard high-resolution colors.

Examine – This program provides a very simple way of viewing each shape in the shape table as it moves across the screen at various speeds.

Placement – This allows the shapes from a table to be placed on the screen under paddle control. This option is the only way to segment a previously created shape table and to re-create it in a different form or with a different set of shapes.

Shift – This utility performs color shifts on the shapes on the screen. Shapes can also be inverted.

The Delimit program allows you to create a shape dot-by-dot in a manner somewhat similar to that used in the character generator section of the *DOS Tool Kit*. One major advantage over the *Tool Kit* is the ability to define the size of the shape in terms of the number of horizontal and vertical dots. Thus one shape is not limited to the size of a single letter. Of course, another major advantage is the ability to create colored shapes. Delimit is easy to use, and a menu is provided at the bottom of the screen in case you forget the commands or you don't read manuals. Shapes are added to the shape table one at a time.

Once a shape has been added to a table, it is relatively easy to remove that shape by using Delimit in combination with the Placement utility. Have you ever tried to combine or eliminate shapes from a regular Apple shape table by hand? With these two utilities, it is easy. Delimit also provides two modes to store shapes. There is the normal shape-saving mode for objects which require smooth movement, and a space-saving mode for objects which require smooth movement, and a space-saving mode for shapes that can move using larger jumps. In general, I found it quite easy to create fairly complicated shape tables using Delimit.

Viewing, Positioning, And Controlling Color

Examine is a simple utility for viewing the shapes in the shape table. Each shape is shown, one at a time, moving across the screen with nine different combinations of "speed" and "delay." The major problem with this utility is that if you want to see the fifth shape in the table, you must first watch the first four shapes dance across the screen nine times each. If you just want to view a shape table, it is quicker to use the Placement utility rather than Examine.

Placement allows each shape to be placed onto the screen as many times and in as many places as desired. This utility has two major uses. The first is when a background picture is needed in which a single shape is to be displayed several times in different places on the screen. Placement can place any shape anywhere on the screen as many times as necessary, after which the screen can be saved. The second major use of Placement is to edit shape tables. While a shape cannot be removed from a shape table by simply deleting the shape, the shapes you want to keep can be placed on the screen using Placement, and then reassembled into a new shape table using Delimit.

The Shift utility allows a shape's color to be shifted. Shapes can also be inverted.

Fast Draw routines are easy to access from BASIC programs. I found the instructions very clear on the methods available to access *Fast Draw* shape tables from BASIC. The only problem with the documentation is that the three demo programs listed in the manual will not work as is. However, with careful reading of the *Fast Draw* instructions, I was able to correct the demo programs. The only feature that I felt was missing from *Fast Draw* was a way to edit existing shapes. Once a shape was made, it could not be changed or used as the basis of the next shape. Therefore, if you needed a series of similar shapes, you would have to start each one from scratch.

Fast Draw (written by Glen Bredon) is an excellent graphics utility: well-written, easy to use, and well-documented. These procedures are so good that you might want to use them in your own programs. Fortunately, Synergistic Software decided not to copy-protect the diskette, so these routines can be used on any other diskette. However, Synergistic Software requests that you sign a license agreement first. There is no fee for the license agreement. That's fair enough, isn't it?

Micro-Sketcher

Micro-Sketcher is a menu-driven graphics utility for creating high-res color pictures, allowing you to create, display, edit, save, fill, and load tables to create full screens. One thing that makes *Micro-Sketcher* unique is that it allows you to create and save segments of a picture rather than having to work with the entire screen. These picture segments then can be displayed individually or in combination to create the final screen image.

I did find some problems with this package. First of all, full screen means only 256 positions horizontally, while, as we all know, the Apple screen is 280 positions in the high-resolution mode. This means there is a wide black border on the right side of the screen. This creates a problem with the fill routines, which fill out to all 280 positions. If a border is not placed around the screen, the color fill routine can cause some rather undesirable effects. In addition, once a color has been selected and an area filled, that area cannot be redefined with a new color. If a new color is desired, that sketch in the shape table will have to be redone. If you are working with the entire screen and choose the wrong color, you will have to start over or live with the color selected.

There is also no continuous draw capability. All lines are drawn as line segments by defining both end points of the line. This method is known as "rubber banding" in some packages because a flashing line is shown on the screen from the first end point to the current position. When the second end point is chosen, the line becomes solid. This is great for drawing tables, rooms, and buildings, but it is very difficult to draw curved shapes such as circles, letters, trees, etc. There are no circle utilities to create circles, or character utilities to add letters or text to your picture, either.

There is also no "paintbrush mode" such as is found in many packages which would allow you to create interesting effects such as shading, trees, bushes, and so on by using different "paint brushes."

A minor problem is that it is too easy to erase the entire screen with the "X-clear" command. After you've worked hard over a picture, a simple slip of the left hand onto the X key can make you want to bang your head against the wall. A two-key command such as CTRL-X would be far better and safer.

The edit mode of *Micro-Sketcher* is unusual. To edit a shape, the program removes one line at a time from the end of the shape. Therefore, if an error was made at the beginning of the shape, all of the lines must be removed until you get back to the line in error, and then the lines must be redrawn. Also, the edit mode may not remove all of the dots from the screen as it removes the lines. These remaining dots cannot be edited out of the picture with this package. You cannot simply draw over these dots with a black pen, because you can only draw white lines on a black background. Start over with a clean screen.

On the positive side, *Micro-Sketcher* has a fast and very efficient fill routine (written by John Conley) which is capable of handling fairly complex shapes. In fact, the fill routine is much better than those in many other graphics packages.

Except for the X key, the program has good protection against faulty input. The documentation is good, but not as clear as the *Fast Draw* documentation. Up to 32 colors are available for the fill routines, and the author has split these colors into compatible groups to eliminate the problem of color smearing when two colors are placed next to each other.

Another nice feature is the use of game paddles or a joystick to roughly position a point, and the use of the I, J, K, or M keys to disable the paddles or joystick and make fine adjustments.

In general, the *Game Animation Package* is well worth the price for people who would like to write animated games, but who do not know 6502 machine language. The *Fast Draw* routines are worth the price of the package themselves for that purpose. While the documentation is fairly good, it does help to first have an understanding of the way the Apple uses graphics and the graphics screens. However, the shape tables created by these two packages are not the same type of shape tables described in the *Apple Reference Manual* or in some other Apple books.

You must use the routines provided on the *G.A.P.* diskette to be able to draw these shapes. In fact, the shape tables created by the two different methods are not really compatible with each other (or at least I was not able to use them interchangeably). However, since the manual explains how to access these shape tables from BASIC, and since the routines are on the diskette, let's go out and add some animation to our games.

Game Animation Package
Synergistic Software
5221 120th Avenue SE
Bellevue, WA 98006
$49.95

©

Mazogs For Sinclair/ Timex

Arthur B. Hunkins

Mazogs is an excellent, single-player, treasure/ maze game for Sinclair/Timex computers with the 16K memory expander. Its full screen graphics make excellent use of the Sinclair/ Timex capability. *Mazogs* is written largely in machine language and runs immediately upon loading. A review copy loaded reliably on my Timex TS-1000. (The program also runs on 16K Sinclair ZX-81 and ZX-80 with 8K ROM.) It is recorded on both sides of the cassette, and comes with a four-page explanatory brochure. (You'd better read it carefully – this game can get complicated!)

Mazogs has three particularly strong points I'd like to mention: 1) there are three levels of play, from neophyte to highly skilled and self-competitive; 2) there are sufficient options so that various strategies may be tried out and implemented; 3) high score is kept (no maximum "high score" limit exists) so that there is always an incentive to do better.

Mazogs are ugly, threatening little creatures who inhabit the treasure maze and love to devour treasure-seekers. They frequently block the way; if you engage them in battle without a sword (swords are scattered throughout the maze), you have only a 50-50 chance of surviving. Your job is to find the treasure and get back out without being devoured. The maze is huge, and the number of moves to the treasure is anywhere from 120 to over 400.

Prisoners With Blinking Eyes

Most of the play takes place on a local scale (full screen), where you can see only several moves in advance. However, a "view" is always available, which gives a larger perspective in your immediate area. Also accessible is a "situation report" which informs you, among other things, how far you are from your goal (treasure or exit).

Your main allies are the "prisoners," with blinking eyes, locked in the walls of the maze by the ruthless Mazogs. Positioned randomly throughout the maze, they know both the way to the treasure and the way out. When you stop to ask advice, they show you the way (marking the path "THIS WAY"). The only problem is that their memories (or yours?) last only about ten seconds. After that, you are on your own again.

In the two advanced levels of play there are four intriguing features: 1) you get only a specified number of moves, depending on total distance – if you exceed this number, you "die" in the maze; 2) you get points (more moves) for killing Mazogs, etc., and lose points for such things as asking for "views," "situation reports" (even "buying a sword" when in the direst of straits); 3) prisoners die once they help you, and swords can be used only once before disappearing; 4) Mazogs themselves become aggressive and mobile – they jump around and attack, sometimes even in twos and threes. Of course, there are various defenses, described nicely in the instructional brochure. The point is that strategy takes a while to develop; so the game takes skill, invites involvement, and has "staying power." In short, *Mazogs* has the ability to become at least moderately addictive.

Something For Everyone

One of the best features is the graphics display. There is a fair amount of animation (Mazogs, treasure-seeker movement, prisoners' eyes blinking, as well as treasure glittering). Much of this is seen in the opening display, which, with its simultaneous animation, is quite impressive.

At game's end, you have the option of playing another game (any skill level), or of seeing (and exploring) the entire maze, including viewing its solution. A bird's-eye view of the maze takes four full screens, and you can see different parts of it by pressing the directional keys. (The same four keys are used to maneuver your treasure-seeker during the game.) Another option is offered at the beginning: a choice of two ways to enter the maze – from the left or right. An initial "view" displays the options as you prepare to start your journey.

There are numerous details, all nicely done, that add to the pleasure and challenge of the game. For example, there are two alternate keypad directional schemes – one is conceptually clearer, the other is faster. Take your choice; there is something for everyone! As a matter of fact there is only one thing I can think of to criticize about *Mazogs* – and I doubt whether the authors could have done anything about it. The program is a bit slow responding to key-presses (they do automatically repeat if you hold them down).

At $14.95, *Mazogs* is a good value and should furnish many hours of creative entertainment. It's one of the better 16K Sinclair/ Timex games out there. I recommend it.

Mazogs
Bug-Byte Software (England)
Distributed by Softsync, Inc.
P.O. Box 480, Murray Hill Station
New York, NY 10156
$14.95

©

Andromeda For Atari

Larry Isaacs

Andromeda is a game distributed by Gebelli Software Inc. It is written in machine language and requires an Atari 400 or 800 with at least 24K, a disk drive, and a joystick. It is a re-release of an earlier version, and current owners can get the new game by returning their old copy to Gebelli Software.

In *Andromeda* you are in control of the "Andromeda" cell, which has invaded the body of a multi-cellular organism. The object of the game is to keep Andromeda alive as long as possible, scoring as many points as you can.

The field of play is the multicellular organism, which is approximately 18 times larger than the display screen. You direct Andromeda about the screen using the joystick. When Andromeda reaches the edge of the screen, the field scrolls underneath Andromeda to display other parts of the organism. Inside the organism, you will see fat cells, which appear as smiling green faces, and blood vessel cells, which are red four-pointed stars. You can even see moving blood cells within the blood vessels, though they do not figure in the game.

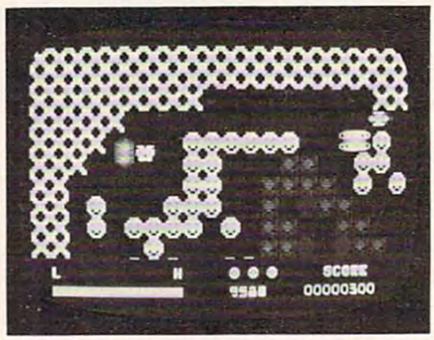
One of the requirements for keeping Andromeda alive is to destroy cells inside the organism. If you fail to destroy cells, Andromeda will become weaker, and could get too weak to move. You destroy a cell by positioning Andromeda just below the cell you wish to destroy and pressing the fire button on the joystick. You may also simply hold the fire button down while you position Andromeda underneath the cells you want to destroy. Destroying cells also scores points. You get 100 points for fat cells, 200 points for a blood vessel cell, and 500 points for one of the few mutant cells in the organism. When a cell is destroyed it will disappear. However, after a certain number of that type have been destroyed, the destroyed cells will begin to reappear in a different color. The new color indicates that this regenerated cell is immune to Andromeda. When most of the cells on the screen are regenerated cells, you will have to move to another part of the organism to seek fresh cells.

The Dread Antibodies

The foes you must face in this game are, naturally enough, antibodies. If an antibody comes in contact with Andromeda, one of Andromeda's three lives is lost. There are four types of antibodies, each with its own pattern of movement. Fortunately, Andromeda is not always at their mercy. Each time you add 5000 points to your score, Andromeda assumes an enlarged state.

If Andromeda comes in contact with an antibody while in the enlarged state, the antibody is destroyed. Naturally, destroying an antibody is worth more points than destroying the regular cells. The points range from 300 to 1000, depending on the type of antibody. Andromeda's enlarged state is only temporary. Fortunately, you are given an audible warning a couple of seconds before Andromeda reverts to its normal size and vulnerability.

At the bottom of the display are several status indicators to assist you during the game. On the left side is an indicator that shows the organism's level of resistance to Andromeda. This level is lowered by destroying cells, and once the level reaches zero, the organism itself is destroyed. On the right side, your score is shown. In the middle is a timer which starts at 9999 and counts down. When you destroy the organism, you get a bonus of the current count times 100 points. Above the timer are indicators, up to three, which show the number of lives Andromeda has left. When Andromeda destroys an organism, Andromeda receives a bonus life and gets to invade another organism.



Dodging antibodies and destroying enemy cells in Andromeda.

Implementation of the game has been carried out fairly well. The use of graphics is simple but good, and player movement is very smooth. The game supports seven levels of play, with the speed of movement and rate of appearance of the antibodies increasing with each level. Andromeda's rate of movement is slower while moving through blood vessel cells. This contributes to the realism within the game.

My major criticism of this product concerns the instruction sheet. The instructions are very sketchy, and in some cases, incorrect. For example, the instructions state that the indicator at the lower left corner shows the level of antibodies. I was unable to see any correlation of this indicator number to the number of antibodies that have been destroyed, or the number present on the screen. Also, illustrations of the different cells in the instructions don't match with what appears on the screen.

Nor do the instructions give any hints on strategy. For example, it isn't very difficult to destroy the organism before the timer count goes below 8000. This means that your bonus score will typically be over 800,000 points. While destroying the organism, you might accumulate about 30,000 points for destroying individual cells and antibodies. As a result, points from destroying individual cells and antibodies become somewhat negligible compared to bonus points. This would seem to imply that destroying the organism as fast as possible is the primary goal. However, the instructions don't even mention that the organism is capable of being destroyed. Fortunately it isn't hard to pick up most of what the game is about just by playing it a few times. However, you could miss out on some subtleties of the game which need a hint or brief description.

One other slight annoyance is that you can't restart the game without reloading it from disk. Since it comes on a copy-protected disk, it would have been nice to eliminate any unnecessary disk wear.

Overall, it is a fairly good game, though not on the level of a *Star Raiders*. If you like the *Pac-Man* style game, you will probably like *Andromeda*. It has a different flavor than *Pac-Man* – you don't have as much control over entering the state when you can eat your opponents, but you also don't have a rigid maze to contend with.

Andromeda
Gebelli Software Inc.
1787 Tribute Road, Suite G
Sacramento, CA 95815
$29.95

©

Shamus For Atari

Tom R. Halfhill, Features Editor

You're prowling along the corridors of yet another unexplored room, searching for the key to the Shadow's lair...

Suddenly you are attacked by a hunting pack of Whirling Drones, Robo-Droids, and the especially deadly Snap-Jumpers. Frantically dodging their molecular disruptors, you hurl several of your contraband Ion-Shivs, blasting them to fragments. Now you're free to pick up the key they were guarding, and you hope that it fits the lock you encountered in that other room far behind you.

But you've dallied too long in this chamber. From out of nowhere descends the Shadow himself, protected by Tri-Gamma body armor impervious to your Ion-Shivs, and he's bent on revenge for the destruction of his henchmen. You break for the exit, but stumble into a wall instead...and instantly disintegrate.

A Blend Of Arcade And Adventure

That's a typical example of how Synapse Software's game *Shamus* is played – and a typical example of how it usually ends as well, since this game is extremely hard to beat. In fact, my guess is that it would take months of frequent play before any mere human could succeed in locating the Shadow's lair and destroying the elusive arch-enemy. This is a game for true addicts.

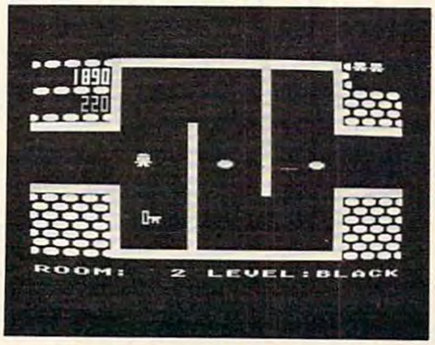
Shamus (pronounced "SHAW-muss" or "SHAY-muss," slang for *detective*) is a one-player game available on disk or cassette which requires at least 16K RAM and a joystick. Programmed by William Mataga, *Shamus* combines the puzzle-solving and exploration features of a graphics adventure game with the fast action of an arcade-style shoot-'em-up.

The object of the game is to locate the hidden lair of a creature known as the Shadow, and then to destroy him in a final struggle. Locating this lair is not easy. There are four levels of rooms to explore, and the only way to advance to the next level is to find the proper key for the proper lock. The locks and keys are color-coded and scattered throughout the rooms, forcing you to wander around, picking up keys and trying them on the various locks.

To give you some idea of the complexity of this task, each level contains no less than 32 rooms – according to the manual. Actually, in my aimless wanderings, I encountered rooms numbered as high as 37 on one level. This means there could be nearly 150 rooms!

The graphics and sound effects in *Shamus* are beautifully done. The game boots up from the disk or cassette with a very good rendition of the theme tune from the old *Alfred Hitchcock Presents* TV show. You then choose from four degrees of difficulty ranging from "novice" to "expert." The manual is absolutely correct when it states that each degree is significantly harder than the last. After briefly sampling the higher degrees, I stuck with "novice" and still found myself outmatched.

You start off in Room 0 on the first level. Your joystick controls a little man in a fedora (after all, what kind of detective would you be without a brimmed hat to pull down over your eyes?). Each of the 32 (or 37, or whatever) rooms on each level occupies a full TV screen. To move to another room, you simply head for a door and walk (or run, as is frequently the case) off the screen. Instantly, the next room appears.



Going for the key to the next level while dodging Whirling Drones in Shamus.

Shamus uses several redefined character sets, and the graphics are among the best I've seen on the Atari. Joystick response is instantaneous, and very often a half-dozen or more multicolored objects will be moving around at once.

These objects, by the way, are the Shadow's henchmen. Searching for keys and locks in scores of rooms spread over four levels would be hard enough, but these creatures are always there to make your life even more difficult. The easiest to dispose of are the Whirling Drones, little pinwheel-shaped machines that home in on your presence. The Robo-Droids aren't too bad either, although they're a headache when attacking in droves with the Whirling Drones. Far more dangerous are the Snap-Jumpers, shifty little critters who move in short leaps in the blink of an eye.

If any of these henchmen shoot you with their molecular disruptors, or even touch you, it's goodby. Your main defense is your inexhaustible supply of Ion-Shivs (Ionic-Short High Intensity Vaporizers). You can throw these in any direction by pressing the fire button while aiming the joystick, and they'll disintegrate anything. Another defense is dodging or even fleeing, but watch out – if you brush against a wall, you'll be instantly zapped to atoms.

By far the most dangerous obstacle, though, is the Shadow himself. If you stay in one room too long – say, half a minute or so – he appears out of nowhere and tries to destroy you with his deadly touch. Since the Shadow wears Tri-Gamma body armor, your Ion-shivs will not kill him. However, they will stun him for a second or two, making escape at least possible.

No Rest For The Weary

Running randomly from room to room spreading wanton destruction doesn't do much good either, since the rooms are repopulated with henchmen as soon as you leave. And they're always positioned between you and the next doorway, or else guarding a lock or key if one is present. This makes your mission a never-ending battle against relentless enemies.

As you advance from level to level (assuming you *do* advance), everything speeds up. The manual describes the final level as "insanely fast." I never made it that far, but I'm not skeptical.

There are a few factors in your favor. You start off with several "lives," and your little man is replaced at the spot where he's zapped – you don't have to restart at Room 0 on the first level. You can also accumulate bonus lives by retrieving bubbling flasks found in some rooms, or occasionally by checking out the question marks left as clues in some rooms. A scoreboard awards points for destroying henchmen and clearing out rooms, but apparently the points are for measuring your progress against other games; they don't seem to win you extra lives or otherwise affect the current game. Although *Shamus* is a one-player contest pitting you against the computer, the manual recommends that two people participate – one to work the joystick and fight the henchmen, and another to keep track of the room layout and locations of locks and keys.

As a final twist, the manual mentions "pod rooms" which exist in another dimension, accessible only through a small "time window." On several occasions I encountered one of these portals, but was never able to pass through.

Overall, I found *Shamus* an exceptionally high quality game, very addicting, and more difficult than most. The programming is top-notch. The only feature I missed was some sort of "pause" option in case the phone rings or the neighbor's house starts burning down. But since the challenge of *Shamus* depends on not giving you time to puzzle out the arrangemment of the rooms, a pause key would make it too easy to cheat. Since there is also no way to save games in progress until later, *Shamus* becomes a test of endurance as well as of memory, cleverness, and reflexes. It succeeds in combining some of the best qualities of arcade and adventure games.

Shamus
Synapse Software
5327 Jacuzzi St. Suite 1
Richmond, CA 94804
$34.95

Moptown – Educational Games For Apple

Sheila Cory

Colorful blocks of varying sizes and shapes that are found in many elementary school classes are called Attribute Blocks. They are used to stimulate rational thinking by giving children experience in distinguishing attributes and carrying out logical operations.

The "Moppets" who live in *Moptown* are computerized Attribute Blocks, with each of the inhabitants identifiable by their peculiar combination of traits.

The traits used to identify the 16 Moppets who populate *Moptown* are: (1) tall or short, (2) fat or thin, (3) red or blue, and (4) Bibbit or Gribbit. All Bibbits have big noses and big feet, and all Gribbits have tails. Like work with Attribute blocks, games in *Moptown* involve logical thinking. Working with attributes on the computer allows, among other things, the random assignment of the attributes, feedback as to the correctness of response, and immediate reinforcement.

Moptown

Moptown is a program designed for elementary school-aged youngsters. Programmed by Leslie Grimm (whose excellent programs – *Bumble Plot*, *Bumble Games*, and *Juggles' Rainbow* – were reviewed in **COMPUTE!** recently), this set of programs consists of 11 different games that develop the ability to identify and isolate attributes. The games are carefully sequenced from easy to hard, providing an ideal structure for understanding and learning. The programs would be appropriate for use in kindergarten through grade six, with the most difficult even providing challenging fun and learning for children in junior high school.

Recognition Games: Easy To Difficult

In Make My Twin, the simplest of the games, the user looks at a *Moptown* villager (or Moppet, as they're called), and then describes its four attributes in order to make its twin. To do this, the child needs to be able to separate each of the attributes from the whole – an excellent activity for the development of analytical thinking. To save typing, the program allows the child to use a one-letter input to describe the attribute. With young children, this can be a very important feature in a program, yet one that some programmers forget to consider.

Who's Different? lines four Moppets up assembly-line style and asks the user to find the one that is different. After identifying the different Moppet, the user then must identify which of the four attributes makes it different. Another possibility in this game is to have four different Moppets drawn, and have the user choose which one is *most* different. This variation is considerably more difficult than the previous one.

What's the Same? is similar to the previous game, except the object is to find the one attribute the Moppets have in common. As in the other games, no help is given if the user continually selects the wrong answer. This could be a problem for a child who chooses to play a game that is beyond his or her level of skill.

"Moppets" line up for review in the Moptown Parade game.

Who Comes Next? is a pattern recognition game. There are three possible patterns: ABABAB, ABBABB, or AABAAB. Four Moppets are lined up; the user determines the pattern and then describes what the fifth Moppet should look like. The task involves not only identifying the pattern, but also dissecting the appropriate Moppet into its four attributes in order to describe them. If the Moppet is described incorrectly, it is drawn the way it was described and the user again has an opportunity to describe the Moppet correctly.

User-Determined Patterns

The next game is Moptown Parade. Like all of these games, it is introduced with an appropriate picture and song – in this case, "She's a Grand Old Flag!" The object of this game is to create the participants in a parade according to a rule determined by the user.

The rule establishes how many traits each successive Moppet in the parade should have that are different from those of the previous Moppet. For example, if the rule is "1", then the next Moppet in the parade will differ from the Moppet in front of him by just one trait. If Moppet 1 is tall, blue, fat, and a Gribbit, then Moppet 2 could be tall, red, fat, and a Gribbit. If a mistake is made, the incorrect Moppet is drawn and then erased so the user can try again.

Who's Next Door? makes trait analysis of two Moppets an essential step for determining the second Moppet of another set. The first pair of Moppets are compared to see which single trait is different. A third Moppet is shown, and its pair must be described so that the two differ in the same attribute as the first pair.

In My Secret Pal, the user selects four traits to describe a

Moppet. The program responds by drawing the Moppet described, and then telling how many of those traits are correct to describe the secret pal. This game is quite a challenge, as the program does not tell you *which* traits are correct, only *how many* are correct. It is up to the user to develop good guessing strategies!

Careful trait analysis is necessary to be successful in the next game, Change Me!. In this game, four boxes are drawn on the screen. A Moppet is drawn in box one and box four. Again, as in Moptown Parade, a rule of "1" or "2" determines how many trait differences there should be in each successive Moppet. The problem is to determine what the second and third Moppet should look like in order for the fourth Moppet to have just the specified number of different attributes.

Clubhouse is more difficult still, requiring logical deductions to decide which Moppet can join the Moppets Club. Each time a Moppet is described, the program responds by telling whether or not he can join the club. The object of the game is to figure out what rule or rules are being applied to each Moppet to either accept him into or reject him from the club.

The last two programs, Moptown Map and Moptown Hotel, carry the skills developed in the previous games a step further. In both of these games, the user has to be concerned with attributes shared by Moppets in the same row *and* the same column. Thinking of relationships in two dimensions makes these two games substantially more difficult than the previous ones; but with mastery of the earlier games, these should be challenging enough to be interesting, yet easy enough to be fun.

Color Monitor Crucial

The documentation for *Moptown* is clear and concise. I disagree with the claim that these programs are suitable for use with a black and white monitor, however. Color is crucial to these programs, as it is one of the four attributes by which the Moppets are distinguished from each other. As the manual states, it *is* possible to discern the differences on a black and white monitor, but I feel it makes the games too difficult. One outstanding feature of the manual is the inclusion of suggestions on how to use these programs when there is just one computer for a whole class.

Sound adds a lot to this program. However, sound can be a distraction in some classroom situations. The program does not have a "sound/no sound" option, which might make it inappropriate for some classes. The program also makes different sounds when a child gets an answer correct than when he or she gets an answer wrong. Some children could be very upset about having others know how they're doing when they're working so hard to master a difficult concept.

How Children Rate *Moptown*

Because it is difficult for me to assess how kids would respond to a program, I gathered a group of "kid consultants" to test out these programs. Bret, 11 years old, spent about two hours on *Moptown*. He said he enjoyed all the games, but felt his friends would most enjoy Moptown Hotel, which is the most difficult. He said he would like to borrow the programs from me in order to have more time with them.

Cara, ten years old, enjoyed all of the programs *except* Moptown Map and Moptown Hotel, which she felt were too difficult. She had only a little more than an hour to spend on the programs, so she would possibly enjoy those difficult ones more if she could work with the games a bit longer. Cara felt her friends would enjoy Clubhouse the most. Like Bret, she asked if she could borrow the diskette for more work with these programs.

Chrissa, eight years old, loved the games. She thought Make My Twin was a little boring because it was too easy, but enthusiastically endorsed Clubhouse. The Kids all tended to ask adults how to play the games rather than read the instructions. In a classroom situation, it would be a good idea for the teacher to introduce each of the games to the whole class before having the children play individually.

Moptown runs on an Apple II Plus with 48K. It comes on diskette, with back-up diskette and manual included in a handy package.

Moptown
Apple Computer, Inc.
20525 Mariani Avenue
Cupertino, CA 95014
$50

VIC Searcher

Heinz Wrosch

There are times when you just can't find something in a large program. Instead of reading every line and wasting your time, why not join the growing number of people who say: "Let the computer do it. It does it better."

This short program is a "BASIC loader" which means that it's written in and can be used in BASIC, but is actually a machine language program. Those DATA statements represent the various instructions (coded as numbers) that the computer can read even more easily and faster than it can follow BASIC instructions. Machine language is, after all, the computer's native tongue.

So Much Faster

You don't have to know machine language to use this handy tool. Tools like "Searcher" are often called *utilities* which means "programs that help you program." For Searcher, all you need to do is to type in the mysterious program (SAVE it for future use) and then any time it's RUN it will figure out where your VIC's highest free memory area is, put itself up there, and build a wall around itself by telling the VIC a white lie: that there is a tad less memory available than there really is.

Finally, it self-destructs using NEW in line 60. *It* does all this, *you* just type LOAD and RUN. That's one good reason why computers should, quite often, do things for you. So much faster.

Now for the fun part. LOAD and RUN your Searcher. It will print a number on the screen which is the address you are going to send the computer to. On many VIC's this will be 7547, but it depends on how much memory your VIC has. Whatever the number is, make a note of it. Now LOAD in some long program. Imagine that you want to remove all the REM statements to save memory. To search them out, type a new BASIC line number into the program at line zero:

0:REM (hit RETURN)

following the zero with a colon and then the thing you want to search for (in this case "REM"). Then directly on the screen (not in a BASIC program) type: SYS 7547 (or whatever number the program told you to use). Instantly you'll have a list of all the places where your REM's appear in the program.

Searching is often useful in *debugging* programs (getting them to work right). You might need to know where all the examples of A$ are in a program, or where all the FOR/NEXT loops are, or something else. You can make adjustments more easily to the entire program if you know where and how often things are used. Or you might decide to change all the occurrences of the name "Tom" to "Sam" or something in a long series of DATA statements.

There are many ways to benefit from Searcher. Add it to your toolbox of VIC utilities and then the next time you need to analyze or modify a long program, to save memory space, or to remove or change a name in your address book program – let the computer do it.

```
10 T=PEEK(55)+256*PEEK(56):CS=0
20 T=T-133:TL=(T/256-INT(T/256))*256:TH=INT(T
   /256)
30 POKE55,TL:POKE56,TH
40 FORI=TTOT+132:READA:POKEI,A:CS=CS+A:NEXTI
50 IF CS<>14881 THEN PRINT"ERROR IN DATA STAT
   EMENTS":STOP
60 PRINT"{CLEAR}SYS";T;"TO START":NEW
100 DATA 160, 0, 177, 43, 133, 1, 200, 177, 43
   , 133, 2
110 DATA 160, 0, 177, 1, 208, 1, 200, 177, 1, ~
   208, 1
120 DATA 96, 160, 0, 177, 1, 141, 52, 3, 200, ~
   177, 1
130 DATA 141, 53, 3, 200, 177, 1, 133, 99, 200
   , 177, 1
140 DATA 133, 98, 200, 24, 165, 43, 105, 5, 13
   3, 67, 166
150 DATA 44, 144, 1, 232, 134, 68, 177, 1, 240
   , 55, 162
160 DATA 0, 193, 67, 240, 4, 200, 24, 144, 242
   , 192, 0
170 DATA 240, 10, 136, 230, 1, 208, 2, 230, 2,
   24, 144
180 DATA 242, 160, 0, 177, 67, 240, 8, 209, 1,
   208, 218
190 DATA 200, 24, 144, 244, 169, 35, 32, 210, ~
   255, 166, 99
200 DATA 165, 98, 32, 205, 221, 169, 32, 32, 2
   10, 255, 173
210 DATA 52, 3, 133, 1, 173, 53, 3, 133, 2, 2
   4, 144, 134
```

©

SuperFont Plus

John Slaby

You can generate excellent Atari game graphics by using ANTIC modes 4 and 5. This program provides an ANTIC version of SuperFont. Requires 16K RAM.

After typing in "SuperFont" (**COMPUTE!**, January 1982), I was very pleased. I couldn't imagine needing any additional functions or purchasing any font that could possibly improve upon it. Then I bought *De Re Atari*, and everything I had read previously in the *Hardware Manual* on ANTIC modes 4 and 5 fell into place. At the same time I realized that it was ANTIC mode 4 that allowed the great graphics in *Caverns Of Mars*. I realized I *could* make some useful additions to the original program. Therefore, I offer SuperFont Plus.

Mr. Brannon stated in his article that it would be easy to expand the program, so I did. The additional commands are the ANTIC, PRINT, and Color Change modes. Of these, only the PRINT mode can be used along with the original version of graphics modes 0, 1, and 2. This expanded version is about 40% longer and, if you only have 16K RAM memory, some manipulation will be required; but you can have an ANTIC version of SuperFont. For those of you that already have SuperFont, just add lines 10, 20, 1601 through 1606 and all lines after and including 2000. Also note the changes in lines 100 through 120, 270, 320, 340, 390 through 400, 650, 1300, 1360, 1370, and 1400 through 1410. Once you do this, you will have the capabilities of designing your own ANTIC 4/5 character set.

For those of you with only 16K, there is a way out. You will have to end up with two fonts: one font, the original, for the Basic-supported graphics modes, and one for the ANTIC 4/5 graphics modes. If you delete the following commands and change lines 250 and 300 to say RAM-4 instead of RAM-8, you will have a functional font. The deleted commands which have limited use for ANTIC 4/5 are: RESTORE (920-930), OVERLAY (870-910), GRAPHICS (1370-1390), WRITE DATA (1290-1360), and QUIT (1130-1140).

Original SuperFont

Here's a quick review of the original SuperFont commands:

EDIT: The character you select via the joystick and pressing of the trigger is copied to the grid in the upper section of the screen. The cursor is relocated to this grid, and you can instantly modify the character by moving the joystick and pressing the trigger to either set or remove a point, as desired.

RESTORE: This will copy the pattern from the first character set to the second, located in the lower half of the screen.

COPY FROM: Select a character which will be copied to the current one you are working on.

COPY TO: The current character will be copied to the selected place.

SWITCH: Exchanges the current character for the one selected.

OVERLAY: Adds the selected character's pattern to the current one.

CLEAR: Clears the pattern of the current character. A must for ANTIC 4/5.

INVERT: Turns current character upside down.

SAVE FONT: Saves character set to disk or tape. Answer "Filename" with either C: or D:filespec. If you see an error message, press any key to return to the menu.

LOAD FONT: Retrieves a character set that you saved. Answer "Filename" like SAVE FONT.

CURSOR-UP or SHIFT DELETE: The line of points the cursor is on is deleted, and the following lines are pulled up to fill the gap.

CURSOR-DOWN or SHIFT INSERT: A blank line is inserted on the line the cursor is in, and all lines below it move down one. The bottom line is lost.

SCROLL LEFT: The bit pattern of the character is shifted left.

SCROLL RIGHT: The bit pattern of the character is shifted right.

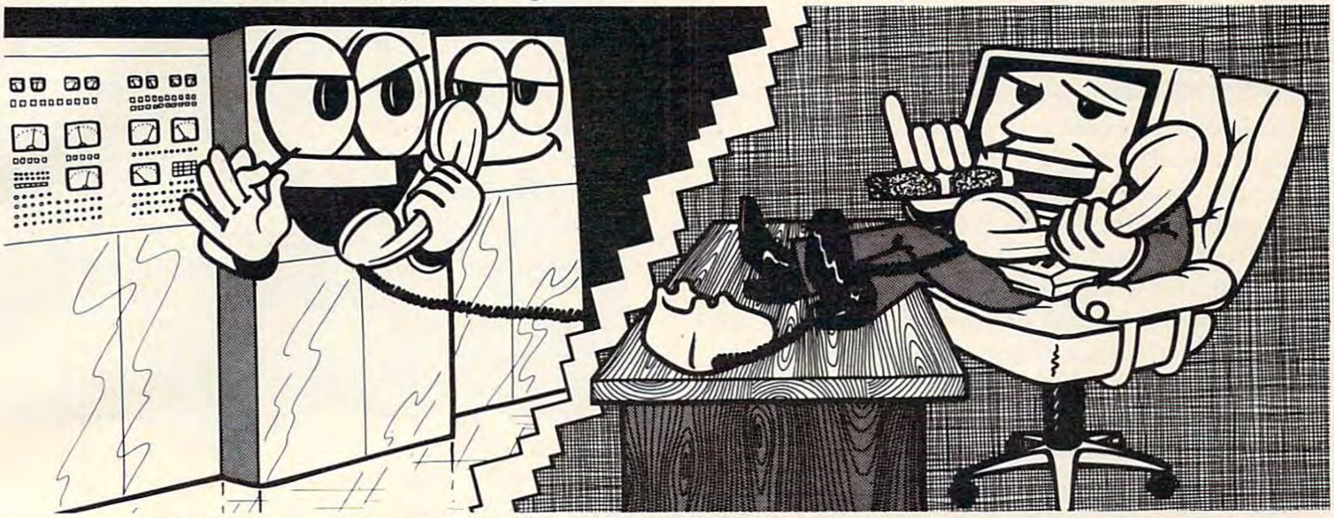
WRITE DATA: The internal code (0-127) of the character and the eight bytes that make it up are displayed in the menu area. Press any key to return to menu.

GRAPHICS: This toggles the TEXT/GRAPHICS option of graphics modes 1 and 2 to let you see each half of the character set.

REVERSE: All blanks become points, and vice

DON'T ASK PROVIDES THE MISSING LINKS

the link between your modem and the outside world. For hassle-free communications, phone right in with TELETARI, The Friendly Terminal.



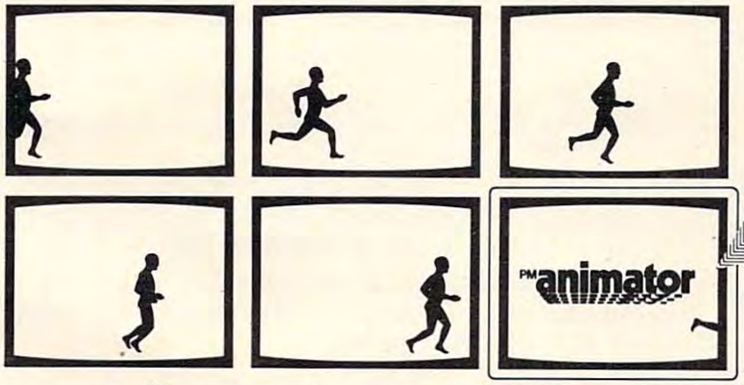
Your Atari has never had such easy access to the whole world of telecommunications – bulletin boards, news reports, large time-sharing computers, the works. Now it's a snap to tap into all these, and it's just as easy to transfer your program or text files to and from a remote computer. Meet TELETARI, The Friendly Terminal. It's just what your modem needs: a powerful, adaptable telecommunications package that's a cinch to use. With TELETARI, you simply choose the desired communications function from a menu. Commonly used terminal parameters are included in the program, but you can change them to suit your needs with a couple of keystrokes, using another handy menu, and store the ones you plan to use again. TELETARI's generous buffer stores up to 20K, so you can review, print, or save received information long after you've hung up the phone. You never knew using a modem could be so convenient. Because it's very flexible, TELETARI is compatible with most modems and a wide variety of computers. And because it works through the RS 232 port, TELETARI is not limited to modem/telephone uses. Put it to work in any RS232 application your imagination can devise – even operating a laser disk!

- buffer of up to 20K
- menu-driven
- highly adaptable
- supports all 850 options
- compatible with 1200 baud modems and BiT 3 Full-view 80™ board
- suitable for any RS232 application

$39.95 Requires Basic, 32K RAM, disk, 850 Interface

---

the link between BASIC and arcade-style graphics. Draw and animate pictures for your own BASIC games and other programs with pm ANIMATOR. Create running men, flying rockets, moving figures of all kinds.



Coming soon from DON'T ASK.

BASIC programmers, pm ANIMATOR puts the power of Player-Missile Graphics at your fingertips.

$34.95 Requires 32K RAM, BASIC, disk.

PM animator

To order direct from Don't Ask, send a check or money order, or call to order COD. Add $2.00 for shipping and handling. California residents add 6% sales tax (6.5% if you reside in L.A. County).

---

the link between fast game action and verbal learning:

WORDRACE

Kids and adults, increase your vocabulary while you compete in this exciting word game.

Disk version:
3 levels of play – Beginner, Regular, Challenge
Requires 32K RAM, disk, BASIC. $24.95

Cassette version:
2 levels of play – Beginner, Intermediate
Requires 16K RAM, cassette, BASIC. $19.95

turn WORDRACE into a history game or a famous athletes game, and get more vocabulary words, with the WORDRACE accessory disk: CLAIM TO FAME/SPORTS DERBY. 3 new games in all.

Disk only. Requires WORDRACE disk. $19.95

the link between you and what your Atari is really thinking:

ABUSE

the insult-exchange program.
Have you cursed out your computer? Now it can understand you and answer back!
Requires 40K RAM, BASIC, disk. $19.95
Release your aggressions! Inflict ABUSE on anyone who's got it coming!

DON'T ASK
the link between technical excellence and the fun of computing. Why do we give you so much? Don't Ask.

DON'T ASK COMPUTER SOFTWARE INC.

2265 Westwood Bl., Ste. B-150
Los Angeles, CA 90064
(213) 476-4583 or 397-8811

versa. Works the same as pressing the Atari logo key and then typing.

QUIT: Exit program.

SuperFont Plus: Three New Commands

The ANTIC(A) command mode modifies the display list so that the lower section of the screen now becomes ANTIC mode 4 except for the last line, which is ANTIC 5. Press A again to return to the original graphics 0, 1, and 2. Once you activate this command, the character set will become mostly unrecognizable. This is because the characters are now four pixels wide instead of eight, but the overall displayed width remains the same. This loss of resolution is the price you have to pay for the multicolor ability of these ANTIC modes.

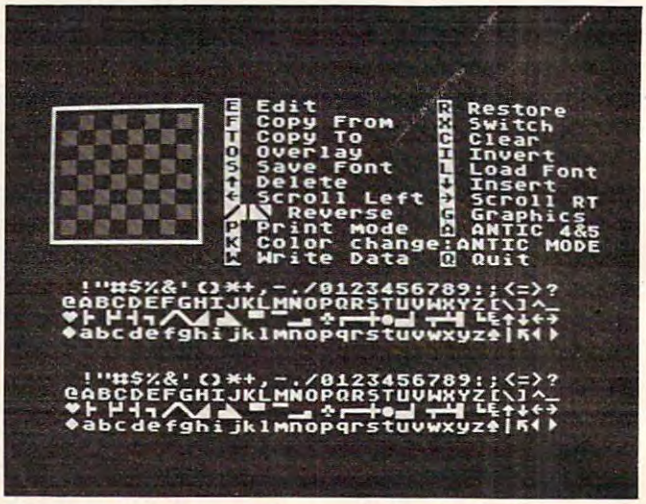
Use all other commands as before; they will work. Please note that the grid now has double-wide pixels when compared to the first display. This is because that binary number you place in each pixel determines the color that will be displayed and you need two bits per color. The binary number is related to the color registers as follows: 00 = Background; 01 = Playfield 0; 10 = Playfield 1; and 11 = Playfield 2. To use Playfield 3's color, you also use binary 11, but the internal code must be 128-255. This is accomplished by using reversed characters via the Atari logo key. There is no way to use this key in any of the original commands, so the PRINT command was created.

The PRINT mode (P) allows you to print any character in the bottom window next to another one just as in normal typing. This mode allows you to see that third playfield color via the logo key. You can type as long as you like, but if you exceed 38 characters, the first one will be lost and all the others will shift left. As noted before, this command can be used with the original graphics 1 and 2.

Since the keyboard is used for typing, the START and SELECT buttons will, respectively, return you to the menu and clear the typing area. When you return to the menu, the typing area isn't automatically cleared; this allows you to work on more than one character at a time, i.e., three characters together as a car, etc. This mode is also useful to get a full screen effect for one line of modified characters.

The final new command is the Color Change mode (K). When I started working with the first two new commands, it became obvious that the ability to change the color of the character I was working on would be very useful. Thus I expanded the Display List Interrupt to give me that ability and added a second interrupt for the background color change.

When you activate this command, you will be able to change only the colors for the ANTIC 4/



The menu and character fonts ready for editing in "SuperFont +."

5 character set. If you want to change the colors for the original graphics modes, modify lines 170 and 300 as desired. The menu area will be cleared, and you will be given the choice of the playfield or background color you want to change. If you change the background, it will affect only the typing window area. I did this to keep the clarity of the character set at its best, and you will probably want to see the change for only one or two characters at a time.

After your register selection, you will be asked for the color and luminosity value (0-14) you want. To help you, a list of colors will be supplied in the menu area. If you give a bad input, you will be asked to try again, starting with the color value. To get the decimal value being used by that register, press R when being offered the color registers and then select a register.

That covers everything; now you should be able to generate some excellent graphics characters like those in *Caverns of Mars* and *Eastern Front*.

The author will make tape copies of the program for those not wanting to type it in themselves. Send a cassette, an SASE mailer, and $3 to:

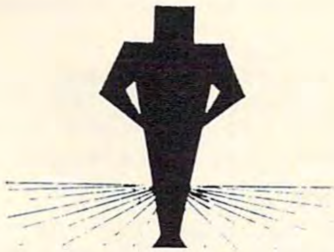
John Slaby
3328 Kaywood Drive
Easton, PA 18042


```
10 GOTO 100
20 POKE 82,14:POSITION 14,0:FOR I=ST
   TO ED:? "{25 SPACES}":NEXT I:RETURN
100 REM *** SUPERFONT + ***
105 REM Character Set Editor
106 REM original
110 REM 11/10/81 Charles Brannon
115 REM ANTIC,COLOR, AND PRINT MODES
120 REM BY John Slaby 8/22/82
140 DIM I(7),FN$(14),N$(3)
150 IF PEEK(1536)=0 THEN GOSUB 1400
160 GRAPHICS 0:POKE 752,1
170 SETCOLOR 2,7,2:SETCOLOR 4,7,2
180 DL=PEEK(560)+256*PEEK(561)+4
```

```
190 SD=PEEK(88)+256*PEEK(89)+12*40:AS
    D=SD+5*40
200 A1=1630:FUNC=1631:A2=1632:LOGIC=1
    628
210 RAM=PEEK(106)-8:PMBASE=RAM*256
220 CHRORG=57344
230 POKE 559,46:POKE 54279,RAM
240 POKE 53277,3:POKE 53256,3
250 CHSET=(RAM-8)*256
260 POKE DL+23,6:POKE DL+24,7
270 POKE DL+17,130:POKE DL+18,112
280 POKE 512,0:POKE 513,6
290 POKE 54286,192
300 POKE 1549,RAM-8:POKE 1672,RAM-8:P
    OKE 1538,0
310 A=USR(1555,CHSET)
320 P0=PMBASE+512+20:P1=PMBASE+640+20
    :P2=PMBASE+768+20:P=PMBASE+896+20
    :T=85:GOSUB 330:GOTO 350
330 FOR I=0 TO 7:FOR J=0 TO 3:T=255-T
    :POKE P0+I*4+J,0:POKE P1+I*4+J,T:
    T=255-T
340 POKE P2+I*4+J,T:NEXT J:T=255-T:NE
    XT I:RETURN
350 POKE 53248,64:POKE 53249,64:POKE
    53250,64
360 POKE 704,198:POKE 705,240:POKE 70
    6,68
370 POKE 53256,3:POKE 53257,3:POKE 53
    258,3:POKE 623,1
380 ? " {Q}{8 R}{E}":FOR I=1 TO 8:? "
    |{8 SPACES}|":NEXT I:? " {Z}{8 R}
    {C}"
385 GOSUB 390:GOTO 490
390 POKE 82,14:POSITION 14,0
400 ? "E Edit{8 SPACES}R Restore"
410 ? "F Copy From{3 SPACES}X Switch"
420 ? "T Copy To{5 SPACES}C Clear"
430 ? "O Overlay{5 SPACES}I Invert"
440 ? "S Save Font{3 SPACES}L Load Fo
    nt"
450 ? "{ESC}{DEL LINE} Delete
    {6 SPACES}{ESC}{INS LINE} Insert"
460 ? "{ESC}{CLR TAB} Scroll Left
    {ESC}{SET TAB} Scroll RT"
470 ? "{F}{B}{E} Reverse{3 SPACES}G G
    raphics"
475 ? "P Print mode  A ANTIC 4&5"
477 ? "K Color change:ANTIC MODE"
480 ? "W Write Data  Q Quit":RETURN
490 FOR I=0 TO 3:FOR J=0 TO 31:POKE S
    D+J+I*40+4,I*32+J:POKE ASD+J+I*40
    +4,I*32+J:NEXT J:NEXT I:?
500 POKE 82,2:POSITION 0,0
510 OPEN #2,4,0,"K:"
520 P=PEEK(764):IF P=255 THEN 520
530 IF P=60 THEN 520
540 IF P=39 THEN POKE 764,168
550 GET #2,K
560 IF K<>ASC("E") THEN 790
570 GOSUB 1750
580 FOR I=0 TO 7:A=PEEK(CHSET+C*8+I):
    FOR J=0 TO 3:POKE P0+I*4+J,A:NEXT
     J:NEXT I
590 POKE ASD+169+(ANTIC*10),C:POKE AS
    D+190+(ANTIC*30),C
600 JX=0:JY=0
610 POSITION JX+4,JY+1
620 ? CHR$(32+128*FF);"{LEFT}";:FF=1-
    FF
630 IF STRIG(0)=0 THEN 750
640 IF PEEK(764)<255 THEN ? " ";:GOTO
     520
650 ST=STICK(0):IF ST=15 THEN 610
660 IF STRIG(0) THEN FOR I=0 TO 100 S
    TEP 20:SOUND 0,100-I,10,8:NEXT I
670 POSITION JX+4,JY+1:? " ";
680 JX=JX+(ST=7)-(ST=11)
690 JY=JY+(ST=13)-(ST=14)
700 IF JX<0 THEN JX=7
710 IF JX>7 THEN JX=0
720 IF JY<0 THEN JY=7
730 IF JY>7 THEN JY=0
740 GOTO 610
750 POKE A1,PEEK(CHSET+C*8+JY):POKE A
    2,2^(7-JX):POKE FUNC,73:A=USR(LOG
    IC)
760 POKE CHSET+C*8+JY,A:FOR J=0 TO 3:
    POKE P0+JY*4+J,A:NEXT J
770 FOR I=0 TO 10:SOUND 0,I*4,8,8:NEX
    T I:SOUND 0,0,0,0
780 GOTO 650
790 IF K<>ASC("F") THEN 830
800 S=C:GOSUB 1750
810 FOR I=0 TO 7:A=PEEK(CHSET+C*8+I):
    POKE CHSET+S*8+I,A:NEXT I
820 C=S:GOTO 580
830 IF K<>ASC("T") THEN 870
840 S=C:GOSUB 1750
850 FOR I=0 TO 7:A=PEEK(CHSET+S*8+I):
    POKE CHSET+C*8+I,A:NEXT I
860 C=S:GOTO 600
870 IF K<>ASC("O") THEN 920
880 S=C:GOSUB 1750
890 FOR I=0 TO 7:POKE A1,PEEK(CHSET+C
    *8+I):POKE A2,PEEK(CHSET+S*8+I):P
    OKE FUNC,9:A=USR(LOGIC)
900 POKE CHSET+S*8+I,A:NEXT I
910 C=S:GOTO 580
920 IF K<>ASC("R") THEN 940
930 FOR I=0 TO 7:POKE CHSET+C*8+I,PEE
    K(CHRORG+C*8+I):NEXT I:GOTO 580
940 IF K<>ASC("C") THEN 960
950 FOR I=0 TO 7:POKE CHSET+C*8+I,0:N
    EXT I:GOTO 580
960 IF K<>ASC("{R}") THEN 980
970 FOR I=0 TO 7:POKE CHSET+C*8+I,255
    -PEEK(CHSET+C*8+I):NEXT I:GOTO 58
    0
980 IF K<>ASC("X") THEN 1010
990 S=C:GOSUB 1750
1000 FOR I=0 TO 7:A=PEEK(CHSET+S*8+I)
     :POKE CHSET+S*8+I,PEEK(CHSET+C*8
     +I):POKE CHSET+C*8+I,A:NEXT I:GO
     TO 580
1010 IF K<>ASC("I") THEN 1030
1020 FOR I=0 TO 7:I(I)=PEEK(CHSET+C*8
     +I):NEXT I:FOR I=0 TO 7:POKE CHS
     ET+C*8+I,I(7-I):NEXT I:GOTO 580
1030 IF K<>ASC("{UP}") AND K<>ASC("
     {DEL LINE}") THEN 1050
1040 FOR I=JY TO 6:POKE CHSET+C*8+I,P
     EEK(CHSET+C*8+I+1):NEXT I:POKE C
     HSET+C*8+7,0:GOTO 580
1050 IF K<>ASC("{DOWN}") AND K<>ASC("
     {INS LINE}") THEN 1070
1060 FOR I=7 TO JY STEP -1:POKE CHSET
     +C*8+I,PEEK(CHSET+C*8+I-1):NEXT
     I:POKE CHSET+C*8+JY,0:GOTO 580
1070 IF K<>ASC("{LEFT}") THEN 1100
1080 FOR I=0 TO 7:A=PEEK(CHSET+C*8+I)
     *2:IF A>255 THEN A=A-256
1090 POKE CHSET+C*8+I,A:NEXT I:GOTO 5
     80
1100 IF K<>ASC("{RIGHT}") THEN 1130
1110 FOR I=0 TO 7:A=INT(PEEK(CHSET+C*
     8+I)/2)
```

NEW FOR ATARI FROM MMG MICRO SOFTWARE

*NOW — THE **TWO** MOST POWERFUL AIDS FOR ATARI BASIC!!*



BASIC COMMANDER

- **Single key entry file commands**
  - — ENTER "D: — SAVE "D:
  - — LIST "D: — RUN "D:
  - — LOAD "D:
- **Single Key DOS functions from BASIC**
  - — FORMAT a disk — LOCK a file
  - — RENAME a file — UNLOCK a file
  - — DELETE a file — DISK DIRECTORY
- **THREE PROGRAMMABLE KEYS!!**
  - — single keys programmed for your own use, even whole subroutines
- **AUTONUMBER**
  - — automatically generates line numbers for you — speeds program entry 25-75%
- **BLOCK DELETE**
  - — deletes any range of lines instantaneously!
- **RENUMBER**
  - — renumbers lines and all references
  - — extensive error trapping
  - — 3 seconds to renumber 500 lines

MMG BASIC DEBUGGER

- **TRACE through your Basic program**
  - — Single step — TRACE while
  - — TRACE UNTIL — change variables
  - — LIST line numbers executed
  - — examine variables' values
- **Full screen BASIC editing**
  - — scroll up or down by cursor
  - — edit your whole program easily
  - — no more LIST line number ranges
- **Split screen mode**
  - — view two parts of your BASIC program at once, and edit both!
  - — scroll each window independently
- **CROSS REFERENCE**
  - — provides a list of variables and the line numbers in which they are used in your program
- **SEARCH FOR PHRASE**
  - — search your BASIC program for any phrase, command or string of characters; let your computer do the searching for you!

EACH PRODUCT ALONE REQUIRES 16K, AND IS AVAILABLE ON DISK FOR *ONLY $34.95*

Now, the convenience of both powerful utilities together in your Atari *at once*

The Combined Basic Commander and MMG Basic Debugger requires 24K. $74.95

ADDITIONAL PRODUCTS

NECESSITIES

RAM TEST II - The fastest and most thorough memory test available for the ATARI has now been further improved! Tests not only all locations, but also tests the memory addressing system. This all machine language program takes 4 minutes to test 48K. It's the only program that tests the cartridge area of RAM. Good for new 400/800 computer owners, for testing new RAM boards and for use in computer stores to test and pinpoint bad memory locations. Bad memory locations are pinpointed so repair is as simple as replacing a chip!

Requires 8K, Disk or Cassette **$29.95**

DISK COMMANDER II - Just save this program on your BASIC disks and it will autoboat and automatically list all programs for the disk into your screen. Simply run any program by typing a single number.

Requires 16K, Disk Only **$29.95**

TUTORIALS

ASTEROID MINERS - This 50 page book and program provides for a unique intermediate to advanced tutorial. A 32K BASIC game utilizing over 25 players in player-missle graphics, machine language subroutines, a redefined character set, multiprocessing utilizing the vertical blank interrupt interval, and much more! The 50 page book included with the program documents each part of the entire program and contains the fully documented source code for both the BASIC and assembly language parts of the program. Use these routines in your own programs. These examples make it easy!

Requires 32K, Disk or Cassette **$34.95**

GAMES

CHOMPER - An all machine language arcade style game with intelligent monsters. Requires 16K Ram, 1 Joystick and nerves of steel.

Available on Disk or Cassette **$29.95**

BUSINESS/HOME

MAILING LIST - Extremely fast BASIC and machine language program. Each data disk holds over 500 files. Sort on any of 6 fields at machine langugage speed or search on any fragment of a field! Use any size labels or envelopes.

Require 40K, Disk Only **$39.95**

NEW

MMG DATA MANAGER - If you frequently find yourself looking for something, only to find it eventually right under your nose, then MMG DATA MANAGER is for you. Organize virtually anything into a computer-searchable format, and let your ATARI do the hunting for you. MMG DATA MANAGER is the first of a series of business applications from MMG MICRO SOFTWARE, all of which will share the ability to access files created by any of them. This flexible database manager will allow many fields, with machine language sorting, on any field. In addition, you have total control of the structure of your data, allowing you to design a database which you feel most comfortable. A special feature of MMG DATA MANAGER is its ability to select for a given value of any single field, or any combination of values from many fields. You could, for instance, determine who lived in Las Vegas, Nevada, and bought item #3145 from you, and whose last name began with SM, and whose telephone number began with (702) 873-4. You'll never lose track of information again! Multiple print options add to the versatility of MMG DATA MANAGER.

Requires 40K, Disk Only **$49.95**

Available At Your Favorite Computer Store
OR Send a Check or Money Order to:

MMG MICRO SOFTWARE

P.O. Box 131 • Marlboro, NJ 07746

OR CALL

(201) 431-3472

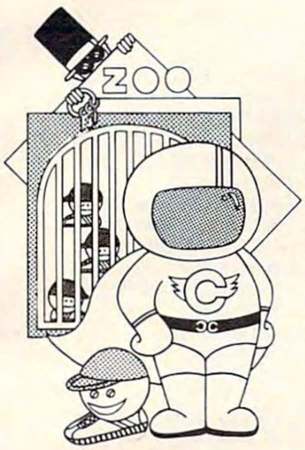
For MasterCard, Visa or COD Deliveries

(Please Add $3.00 For Postage & Handling)

NJ RESIDENTS ADD 5% FOR SALES TAX

```
1120 POKE CHSET+C*8+I,A:NEXT I:GOTO 5
     80
1130 IF K<>ASC("Q") THEN 1150
1140 POKE 53248,0:POKE 53249,0:POKE 5
     3250,0:POKE 53277,0:GRAPHICS 0:E
     ND
1150 IF K<>ASC("S") THEN 1210
1160 GOSUB 1610:POKE 195,0
1170 TRAP 1190:OPEN #1,8,0,FN$
1180 A=USR(1589,CHSET)
1190 CLOSE #1:TRAP 40000:IF PEEK(195)
      THEN 1260
1200 POKE 54286,192:GOTO 580
1210 IF K<>ASC("L") THEN 1290
1220 GOSUB 1610:POKE 195,0
1230 TRAP 1250:OPEN #1,4,0,FN$
1240 A=USR(1619,CHSET)
1250 CLOSE #1:TRAP 40000:IF PEEK(195)
     =0 THEN 1200
1260 POSITION 14,0:? "{BELL}*ERROR -"
     ;PEEK(195);"*"
1270 IF PEEK(764)<255 THEN POSITION 1
     4,0:? "{19 SPACES}":GOTO 1200
1280 GOTO 1270
1290 IF K<>ASC("W") THEN 1370
1300 ST=0:ED=11:GOSUB 20:N$="
     {3 SPACES}":L=LEN(STR$(C)):N$(1,
     L)=STR$(C):L=LEN(N$):POSITION 14
     ,0
1310 FOR I=1 TO L:? CHR$(ASC(N$(I,I))
     +128);:NEXT I:? ">"
1320 Z=0:FOR I=0 TO 2:FOR J=0 TO 1+(I
     >0):A=PEEK(CHSET+C*8+Z):Z=Z+1
1330 SOUND 0,(I*3+J)*10+50,10,8
1340 ? A;",":NEXT J:? "{BACK S}":NEXT
      I:SOUND 0,0,0,0
1350 IF PEEK(764)=255 THEN 1350
1360 GOSUB 20:GOSUB 390:GOTO 520
1370 IF K<>ASC("G") THEN 2000
1380 CF=1-CF:POKE 1549,RAM-8+2*CF
1390 GOTO 520
1400 GRAPHICS 2+16:SETCOLOR 4,1,4:POS
     ITION 5,3:? #6;"SUPERFONT +"
1410 POSITION 5,5:? #6;"patience{3 N}
     ":POSITION 2,11:? #6;"john slaby
     ":POSITION 2,7:? #6;"ORIGINAL by
     "
1415 POSITION 2,8:? #6;"CHARLES BRANN
     ON":POSITION 2,10:? #6;"+   by"
1420 FOR I=1536 TO 1710:READ A:POKE I
     ,A:POKE 709,A:SOUND 0,A,10,4:NEX
     T I
1430 SOUND 0,0,0,0:RETURN
1440 DATA 72,169,100,141,10,210
1450 DATA 141,24,208,141,26,208
1460 DATA 169,6,141,9,212,104
1470 DATA 64,104,104,133,204,104
1480 DATA 133,203,169,0,133,205
1490 DATA 169,224,133,206,162,4
1500 DATA 160,0,177,205,145,203
1510 DATA 200,208,249,230,204,230
1520 DATA 206,202,208,240,96,104
1530 DATA 162,16,169,9,157,66
1540 DATA 3,104,157,69,3,104
1550 DATA 157,68,3,169,0,157
1560 DATA 72,3,169,4,157,73
1570 DATA 3,32,86,228,96,104
1580 DATA 162,16,169,5,76,58
1590 DATA 6,9,104,169,0,9,0,133
1600 DATA 212,169,0,133,213,96
1601 DATA 72,138,72,152,72,169,0,162,
     0,160,0
1602 DATA 141,10,212,141,26,208
1603 DATA 142,24,208,140,25,208
1604 DATA 169,0,141,22,208,141,10,210
     ,169,6,141,9,212,169,0,141,23,20
     8,169,156,141,0,2
1605 DATA 104,168,104,170,104,64,72,1
     69,0,141,10,212,141,26,208,169,1
     04,141,10,210,141,0,2,104,64
1610 POSITION 14,0:? "Filename?";
1620 FN$="":K=0
1630 POKE 20,0
1640 IF PEEK(764)<255 AND PEEK(764)<>
     39 AND PEEK(764)<>60 THEN 1670
1650 IF PEEK(20)<10 THEN 1640
1660 ? CHR$(21+11*K);"{LEFT}";:K=1-K:
     GOTO 1630
1670 GET #2,A
1680 IF A=155 THEN ? " ";:FOR I=1 TO
     LEN(FN$)+10:? "{BACK S}";:NEXT I
     :RETURN
1690 IF A=126 AND LEN(FN$)>1 THEN FN$
     =FN$(1,LEN(FN$)-1):? "{LEFT}";CH
     R$(A);:GOTO 1630
1695 IF A=126 AND LEN(FN$)=1 THEN ? C
     HR$(A);:GOTO 1620
1700 IF A=58 OR (A>48 AND A<57) OR (A
     >65 AND A<=90) OR A=46 THEN 1720
1710 GOTO 1630
1720 IF LEN(FN$)<14 THEN FN$(LEN(FN$)
     +1)=CHR$(A):? CHR$(A);
1730 GOTO 1630
1740 END
1750 REM GET CHOICE OF CHARACTER
1760 CY=INT(MRY/32):CX=MRY-32*CY
1770 C=CX+CY*32
1780 POKE SD+CX+CY*40+4,C+128
1790 POKE ASD+CX+CY*40+4,C+128
1800 IF STRIG(0)=0 OR PEEK(764)<255 T
     HEN MRY=C:GOTO 1900
1810 ST=STICK(0):IF ST=15 THEN 1880
1820 POKE 53279,0
1830 GOSUB 1900
1840 CX=CX-(ST=11)+(ST=7):CY=CY-(ST=1
     4)+(ST=13)
1850 IF CX<0 THEN CX=31:CY=CY-1
1860 IF CX>31 THEN CX=0:CY=CY+1
1870 IF CY<0 THEN CY=3
1880 IF CY>3 THEN CY=0
1890 GOTO 1770
1900 POKE SD+CX+CY*40+4,C
1910 POKE ASD+CX+CY*40+4,C
1920 RETURN
2000 IF K<>ASC("A") THEN 2200
2005 POKE 54286,0
2007 POKE ASD+169+(ANTIC*10),0:POKE A
     SD+190+(ANTIC*30),0
2010 IF ANTIC=1 THEN 2100
2020 POKE DL+24,5
2030 FOR I=19 TO 23:POKE DL+I,4:NEXT
     I:POKE DL+22,132
2040 POKE 512,104:ANTIC=1
2050 COLF0=2*16+6:COLF1=6*16+6
2060 COLF2=10*16+8:COLF3=15*16+8
2070 POKE 1664,COLF0:POKE 1648,COLF1
2080 POKE 1650,COLF2:POKE 1677,COLF3
2090 POKE 54286,192:T=51:GOTO 2127
2100 ANTIC=0:POKE DL+23,6:POKE DL+24,
     7
2110 POKE 512,0:FOR I=19 TO 22:POKE D
     L+I,2:NEXT I
2120 POKE 54286,192:T=85
2127 GOSUB 330:POKE ASD+169+(ANTIC*10
     ),C:POKE ASD+190+(ANTIC*30),C:GO
     TO 520
2200 IF K<>ASC("P") THEN 3000
2205 ST=0:ED=10:GOSUB 20
```

NEXA *presents*



Captain Cosmo is an exciting fast-action video arcade game. It can be played by 1 to 4 players and has 99 skill levels. Try it and you can't let go!

Requires Atari 400/800 with 32K Joysticks, and a Disk Drive.

Delta Squadron is a strategic war game that really puts you in the pilot's seat. With this game you will experience the thrill and excitement of a real space pilot. **Delta Squadron** is a "must" for all strategic game enthusiasts, and a change of pace for those who wants a challenge!

Requires 64K Apple II with DOS 3.3 and paddle.

Superbowl Football is a realistic football game. You can design your own plays and has thousands of defensive and offensive plays. This is the ultimate in computer football games.

Requires Atari 400/800 with 48K, a Disk Drive and Joysticks.

We revolutionize our packaging designs to be convenient, compact, durable, and to protect the diskettes from dust and moisture.

Ask for us at your local stores or your distributor.

NEXA CORPORATION
P. O. Box 26468
San Francisco, CA 94126-6468
(415) 387-5800

CRAM UP TO 90K IN YOUR ATARI

ACA introduces the next generation of memory upgrades. By using simple yet innovative techniques, the cRAM board puts complete control of up to 90K of addressable memory at your fingertips.

Features:

Powers up with 52K of continuous RAM. 4K overwrite protected RAM for assembly object code and BASIC user functions; 52K for boot users.

Cartridge ROM can be replaced by RAM under program control. Simplifies cartridge program development and testing; or 8K/16K hidden RAM for temporary data.

OS ROMs can be replaced by RAM, for OS development or use of an alternate OS.

Built-in flexibility. Several customizing options have been built into the cRAM board. These, along with the features above, convert your Atari 400 into a truly powerful tool.

IN STOCK for immediate shipment.

INTRODUCTORY PRICE

KIT: $169.00
(regular price $189.00)

ASSEMBLED AND TESTED: $209.00
(regular price $229.00)

Applied Computer Alternatives, Inc.

1600 Wilson Boulevard, Suite 900
Arlington, Virginia 22209
(703) 525-6960

DEALER INQUIRIES INVITED.

Requires minor modification to your Atari RAM board. For Atari 400 only.

ATARI 400 is a registered trademark of ATARI, Inc.

```
2210 POSITION 14,0:CT=0
2220 ? "{5 SPACES}PRINT MODE"
2230 ? :? " Press START to return"
2240 ? "{5 SPACES}to menu"
2250 ? :? " Press SELECT to clear"
2260 ? "{3 SPACES}typing area"
2270 KK=PEEK(53279):IF KK=6 THEN GOSU
     B 390:GOTO 520
2280 IF KK=5 THEN 2600
2290 P=PEEK(764):IF P=255 THEN 2270
2300 GET #2,K
2302 IF K>=0 AND K<32 OR K>=128 AND K
     <160 THEN K=K+64:GOTO 2310
2304 IF K>=32 AND K<96 OR K>=160 AND
     K<224 THEN K=K-32
2306 IF CT>(ANTIC+1)*17 THEN 2320
2310 POKE ASD+161+CT,K:POKE ASD+181+(
     ANTIC*20)+CT,K:CT=CT+1:GOTO 2270
2320 FOR I=0 TO 17*(ANTIC+1):POKE ASD
     +161+I,PEEK(ASD+162+I):POKE ASD+
     181+(ANTIC*20)+I,PEEK(ASD+182+(A
     NTIC*20)+I)
2330 NEXT I:CT=17*(ANTIC+1):GOTO 2310
2600 FOR I=0 TO 19*(ANTIC+1):POKE ASD
     +161+I,0:POKE ASD+181+(ANTIC*20)
     +I,0:NEXT I:CT=0:GOTO 2270
3000 IF K<>ASC("K") THEN 520
3010 ST=0:ED=10:GOSUB 20:DIS=0
3020 POKE 82,14:POSITION 14,0:? "COLO
     R CHANGE MODE"
3030 ? " PRESS K TO RETURN"
3040 ? "{5 SPACES}TO MENU"
3050 ? " 0 PLAYFIELD 0"
3060 ? " 1 PLAYFIELD 1"
3070 ? " 2 PLAYFIELD 2"
3080 ? " 3 PLAYFIELD 3"
3090 ? " B BACKGROUND":? "R READ REGI
     STER"
3100 GET #2,K:DIS=0:IF K=ASC("0") THE
     N DIS=18
3105 IF K=ASC("R")THEN RDE=1:GOTO 3100
3110 IF K=ASC("1") THEN DIS=31
3120 IF K=ASC("2") THEN DIS=2
3130 IF K=ASC("3") THEN DIS=4
3140 IF K=ASC("B") THEN DIS=48
3150 IF K=ASC("K") THEN GOSUB 390:GOT
     O 520
3155 IF RDE=1 THEN 3410
3160 IF DIS=0 THEN 3100
3170 ST=2:ED=10:GOSUB 20
3180 POKE 82,14:POSITION 14,0
3190 ? "0 GREY  1 GOLD  2 ORANGE"
3200 ? "3 RED{3 SPACES}4 PINK  6 PURP
     LE"
3210 ? "7 BLUE  8 BLUE  9 LT.BLUE"
3220 ? "10 TURQOUISE 11 GREENBLUE"
3230 ? "12 GREEN{5 SPACES}13 YELLOW/G
     R"
3240 ? "14 ORANGE/GR 15 LI.ORANGE"
3245 TRAP 3400
3250 INPUT COL:?"{3 SPACES}Luminosity"
3260 ? " input(0-14)";
3270 INPUT LUM
3280 CLCHG=COL*16+LUM
3290 POKE 1646+DIS,CLCHG
3300 GOTO 3010
3400 TRAP 40000:POSITION 14,6:? "TRY
     AGAIN":FOR I=1 TO 100:NEXT I:POS
     ITION 14,6:? "{9 SPACES}":POSITI
     ON 14,6:GOTO 3245
3410 RDE=0:DRE=PEEK(1646+DIS):POSITIO
     N 14,9:? "COLOR REGISTER ";CHR$(
     K);"=";;"{3 SPACES}";"{3 LEFT}";
     DRE:GOTO 3100
```

©

Creating Graphics On The Expanded VIC

Ed Harris

This short program will simplify creating new character sets and graphics on an expanded VIC.

Do you have more than 8K of RAM in your VIC? Do you want to make your own character set? Any character sets in RAM must be in the VIC's *internal* memory. When you add the first 8K memory expander, screen memory changes to 4096, and BASIC starts at decimal 4608, leaving no room to put your character set.

This program raises the bottom of memory to 8192 ($2000) and copies the character sets from 32768 to 35839 down into RAM starting at 5120 and going to 8191. You can then create new character sets or game graphics for use on your expanded VIC.

The BASIC program puts the machine code at $3000 and will be erased when you load a program.

All commands still work properly, and you can change from standard to custom characters with "POKE 36869,PEEK(36869) AND 240 OR 13" and restore to normal by "POKE 36869,PEEK (36869)AND 240 OR 0".

```
5 REM **************
6 REM FOR VICS WITH
7 REM  MORE THAN 8K.
8 REM  **************
9 REM  MOVES BASIC TO
10 REM 8192
11 REM *************
12 REM AND COPIES
13 REM CHARACTER SET
14 REM TO 5120-8191
15 REM *************
16 FORT=12288 TO 12379: READ N:POKE T,N:NEXTT
17 PRINT"{CLEAR}SYS12288":FORT=631TO633:POKET
   ,145
18 NEXT T
19 POKE634,13:POKE635,131:POKE198,5:END
20 DATA56,32,156,255,200,24,32,156,255
21 DATA 174,44,0,232,142,44,0,142,46
22 DATA 0,173,46,0,201,32,208,230,169
23 DATA 0,141,0,32,141,1,32,141,2
24 DATA 32,169,205,141,5,144,162,0,142
25 DATA 123,48,174,123,48,189,0,128,157
26 DATA 0,20,224,255,240,7,232,142,123
27 DATA 48,76,47,48,172,55,48,192,31
28 DATA240,14,200,140,55,48,174,52,48
29 DATA 232,142,52,48,76,42,48,96,96,0,0
```

©

Vehicle Cost Performance

Linton S. Chastain

Use your Radio Shack Color Computer to analyze your car's performance. The program is written for systems with 32K, Extended BASIC, and disk capability, but the article notes the necessary changes for use with 16K systems or for non-Extended BASIC systems with cassette.

Have you ever wondered how much your car was costing to operate or whether you were getting good fuel performance? If so, you may want to load the following program and run it.

The program – "Vehicle Cost and Performance" – was written on a 32K Color Computer with Extended BASIC and disk drive. It requires a minimum of 4.813K to load and 12.727K to run, as is. If you have a 16K machine with Extended BASIC and a disk drive, you will have to use "Pclear1" and adjust lines 40 and 50 in order to run the program. Line 40's clear x is 10 times "MR" for safety. I have two years of data and a minimum of two entries per month, and have not run out of storage in memory.

Those of you who have a cassette recorder and non-Extended BASIC will have to make the following changes:

```
130 PRINT"5-READ OLD MASTER FILE":PRINT"FROM C
    ASSETTE
150 PRINT"7-WRITE NEW MASTER FILE":PRINT"TO CA
    SSETTE
670 OPEN"I",#-1,T$:PRINT"READING FILE: ";T$:IN
    PUT#-1,N
680 IFN>MR THEN PRINT"*** TOO MANY FILES ON CA
    SSETTE***":END
700 FORJ=1 TO N:INPUT#-1,A(J),D$(J),O(J),G(J),
    NO$(J),C(J):PRINTJ:NEXTJ
720 CLOSE:GOSUB1440
770 OPEN"O",#-1,T$:WRITE#-1,N
790 FORJ=1 TO N:PRINT#-1,A(J),D$(J),O(J),G(J),
    NO$(J),C(J):PRINTJ:NEXTJ
800 CLOSE:GOSUB1440
```

CHANGE LINE 540 "PRINT @ 192, "STRING$(31,"-")" TO PRINT @ 192, "-------------------------------"

CHANGE LINE 1140 "PRINT @ 320, "STRING$(62,32)" TO PRINT @ 320, "{type 62 spaces}"

CHANGE LINE 1350 "PRINT @ 320, "STRING$(32,32)" TO PRINT @ 320, "{type 31 spaces}"

CHANGE LINE 1580 "PRINT #-2, "STRING$(48,32);" TO PRINT #-2, "{type 48 spaces}";

Those who have Extended BASIC do not have to change anything in lines 540, 1140, 1350, or 1580.

Record Keeping

The program keeps records on Maintenance (accnt #4), Gas (accnt #5), Operating Fees (accnt #6), and Other (accnt #7). It also has two flags that are keyed to Dates. They remind you at least one month in advance of the event's due date, so that you can organize your budget and have time to accomplish the task. These two flags are in record position (J), one and two. You can use these two records to alert you to dates for needed oil changes or to the due dates of your tag, license, inspection, and insurance.

One word of caution about record numbers, "J=1 to N": if you change a record number less than "N", you must enter the last record that is in memory in order to establish the proper "N" again. If you do not enter the last "N", then "N" will become the changed record number.

A third flag, in record position (J) three, keeps track of your vehicle's best MPG performance as well as the date of entry in which it occurred. This flag will flash on your screen if your present entry is equal to one MPG lower than your best MPG (recorded in record three) when you are in the "DISPLAY MILEAGE" section of the program. If you wish to tighten or loosen this criterion, you can change line :1260 IF G(3)-Z=›1 to anything smaller or larger than one.

Making Hard Copies

This program also permits you to make a hard copy of the information generated by the program and/or its data base. You can make a hard copy of "Display Cost," "Display Mileage," "Display Data," and "Display Cost/Mile" by pressing shift down arrow when "PRESS ENTER FOR RETURN" appears on the screen. This action activates line 1540, which in turn activates an eight-line subroutine. The subroutine is very handy if you want a hard copy of text generated by a program. It is incorporated in my "Energy Monitor" program (**COMPUTE!**, August 1982, #27).

The nice thing about this subroutine is that it not only prints out what is on the screen, but it also allows you to control how much paper you wish to waste by controlling "VIM". "VIM" is defined as the last video text memory location which you want outputted to a printer. The Color Computer video memory occupies decimal memory 1024 through 1535. Each of the 16 lines has 31 memory locations plus the first memory of that line. For example, line one contains decimal memory location 1024 through 1055, while line two contains decimal memory location 1056 through 1087, and so forth until line 16.

This subroutine probably can be used on other computers as long as you are aware that the video memory location may be different; the number of each line's memory location and the number of lines may also be different. Try to incorporate the subroutine into your programs. On at least two occasions, it has helped me avoid having to write two different programs, one for the screen and one for the printer.

The second hard copy is generated in the "Display Data" part of the program. By pressing the up arrow, you will dump all records to your printer in nice, neat columns on a 80-character/line printer. This may come in handy if you are selling your car and the buyer wants a fairly complete record of maintenance and cost. However, if you want one or more of the data records, you can selectively print out each one by using shift down arrow, instead of the up arrow, while in the "Display Data" mode.

I hope you will find this program as useful as I have in helping keep track of cost and maintenance problems. It may help you make a more objective decision when purchasing your next vehicle or determining whether keeping your current vehicle might not be more cost effective.

```
10 'VEHICLE COST AND PERFORMANCE
40 CLEAR2000
50 MR=200:N=0
60 DIMA(MR),D$(MR),O(MR),G(MR),NO$(MR),C(MR)
70 CLS:AA=0:AB=0:AC=0:AD=0
80 PRINT"VEHICLE COST AND PERFORMANCE":PRINT:PRINT"COMMAND LIST #1"
90 PRINT"1-DISPLAY COST"
100 PRINT"2-DISPLAY MILEAGE"
110 PRINT"3-DISPLAY DATA"
120 PRINT"4-DISPLAY COST/MILE"
130 PRINT"5-READ OLD MASTER FILE FROM DISK"
140 PRINT"6-INPUT NEW DATA"
150 PRINT"7-WRITE NEW MASTER FILE TO DISK"
160 PRINT:INPUT"ENTER COMMAND BY NUMBER";R:IFR<1 OR R>7 THEN70
170 ON R GOSUB 470,1160,580,1410,650,190,740
180 GOTO70
190 CLS:PRINT:PRINT"ENTER THE FOLLOWING AS REQUESTED"
200 PRINT:INPUT"INPUT N";R:N=R:IFN<0 THEN 200
210 INPUT"ACCOUNT CODE";R:A(N)=R:IFR<0 THEN 210
220 INPUT"DATE (E.G. 07/31/82)";R$:R=LEN(R$):IFR<8 OR R>8 THEN 220
230 D$(N)=R$
240 INPUT"ODOMETER M=MILES OR K=KILOMETERS";R$
250 IFLEFT$(R$,1)="M" THEN 260 ELSE 270
260 INPUT"ODOMETER IN MILES";R:O(N)=R:IFR<0 THEN 260 ELSE 280
270 INPUT"ODOMETER IN KILOMETERS";R:O(N)=R*.62:IFR<0 THEN 270
280 INPUT"FUEL MEASUREMENT G=GAL AND L=LITER";R$
290 IFLEFT$(R$,1)="G" THEN 300 ELSE 310
300 INPUT"AMOUNT OF FUEL IN GAL.";R:G(N)=R:IFR<0 THEN 300 ELSE 320
310 INPUT"AMOUNT OF FUEL IN LITER";R:G(N)=R/3.785:IFR<0 THEN 310
320 INPUT"NOTE";R$:NO$(N)=R$:R=LEN(R$):IFR>10 THEN 320
330 INPUT"COST";R:C(N)=R
340 CLS:PRINT:PRINTTAB(3);"CHECK     N:";N
350 PRINTTAB(3);"    ACC #:";A(N)
360 PRINTTAB(3);"    DATE:";D$(N)
370 PRINTTAB(3);"    ODOMETER:";O(N)
380 PRINTTAB(3);"    FUEL:";G(N)
390 PRINTTAB(3);"    NOTE:";NO$(N)
400 PRINTTAB(3);"    AMT:";C(N)
410 PRINT:PRINT"   -IS INPUT O.K.?-":PRINT
420 INPUT"(Y=YES,N=NO,F=YES AND FINISHED)";R$:R$=LEFT$(R$,1)
430 IFR$="N" THEN PRINT"REDO LAST DATA":GOTO200
440 IFR$="F" THEN RETURN
450 IFR$<>"Y" THEN 420
460 GOTO200
470 VIM=1279:FORJ=4 TO N
480 IFA(J)=4 THENAA=AA+C(J)ELSE490
490 IFA(J)=5 THENAB=AB+C(J)ELSE500
500 IFA(J)=6 THENAC=AC+C(J)ELSE510
510 IFA(J)=7 THENAD=AD+C(J)
520 NEXTJ
530 AE=AA+AB+AC+AD
540 CLS:PRINT@0,"CATEGORY","COST":PRINT@64,"MAINTENANCE",AA:PRINT@96,"GAS",AB
545 PRINT@128,"OPER. FEES",AC:PRINT@160,"OTHER",AD:PRINT@192,STRING$(31,"-")
549 PRINT@224,"TOTALS",AE
550 YR(3)=2:YR(4)=2:MO(3)=2:MO(4)=2:E=0:F=0:GOSUB820
560 GOSUB1440
570 RETURN
580 VIM=1247:K=0:L=0:CLS
590 K=K+1:L=L+1:IFL>N THEN L=N
600 FORJ=K TOL:PRINT"N",J:PRINT"ACCNT",A(J):PRINT"DATE",D$(J):PRINT"MILEAGE",O(J):PRINT"FUEL",G(J)
605 PRINT"NOTE",NO$(J):PRINT"AMOUNT",C(J):NEXTJ:PRINT
610 PRINT@384,"PRESS ↑ TO PRINT TO PRINTER"
620 IFPEEK(341)=247 THEN 1480
630 IFL=N THEN GOSUB1440:RETURN
640 PRINT@416,"HIT ENTER TO CONTINUE":GOSUB1440:CLS:VIM=1279:GOTO590
650 R$="READING":PRINT
660 INPUT"NAME OF FILE";T$
670 OPEN"I",#1,T$:PRINT"READING FILE: ";T$:INPUT#1,N
680 IFN>MR THEN PRINT"*** TOO MANY FILES ON DISK ***":END
690 PRINT"READING RECORDS # ";
700 FORJ=1 TO N:INPUT#1,A(J),D$(J),O(J),G(J),NO$(J),C(J):PRINTJ:NEXTJ
710 PRINTN;" DATA RECORDS READ"
720 CLOSE#1:GOSUB1440
730 RETURN
740 IFN<1 THEN PRINT"*** NO DATA TO WRITE ***":GOSUB1440:RETURN
750 R$="WRITING":PRINT
760 INPUT"NAME OF FILE";T$
770 OPEN"O",#1,T$:WRITE#1,N
780 PRINT"WRITING FILE: ";T$:PRINT"      RECORDS # ";
790 FORJ=1 TO N:WRITE#1,A(J),D$(J),O(J),G(J),NO$(J),C(J):PRINTJ:NEXTJ
800 CLOSE#1:GOSUB1440
810 RETURN
820 YR$(0)=RIGHT$(D$(N),2):YR$(1)=RIGHT$(D$(1),2):YR$(2)=RIGHT$(D$(2),2)
```

```
830 YR(0)=VAL(YR$(0)):YR(1)=VAL(YR$(1)):YR(2)=
    VAL(YR$(2))
840 YR(3)=YR(1)-YR(0):YR(R)=YR(2)-YR(0)
850 IFYR(3)=0 OR YR(3)=1 THEN GOSUB970 ELSE 86
    0
860 IFYR(4)=0 OR YR(4)=1 THEN GOSUB880 ELSE RE
    TURN
870 RETURN
880 MO$(0)=LEFT$(D$(N),2):MO$(2)=LEFT$(D$(2),2
    )
890 MO(0)=VAL(MO$(0)):MO(2)=VAL(MO$(2))
900 MO(4)=MO(2)-MO(0)
910 IFYR(4)=0 THEN 920 ELSE 930
920 IFMO(4)=0 OR MO(4)=1 THEN MO(4)=0 ELSE MO(
    4)=1
930 IFYR(4)=1 THEN 940 ELSE 950
940 IFMO(4)=-10 OR MO(4)=-11 THEN MO(4)=0 ELSE
     MO(4)=1
950 IFMO(4)=0 THEN F=2 ELSE F=0
960 GOTO1070
970 MO$(0)=LEFT$(D$(N),2):MO$(1)=LEFT$(D$(1),2
    )
980 MO(0)=VAL(MO$(0)):MO(1)=VAL(MO$(1))
990 MO(3)=MO(1)-MO(0)
1000 IFYR(3)=0 THEN 1010 ELSE 1020
1010 IFMO(3)=0 OR MO(3)=1 THEN MO(3)=0 ELSE M
     O(3)=1
1020 IFYR(3)=1 THEN 1030 ELSE 1040
1030 IFMO(3)=-10 OR MO(3)=-11 THEN MO(3)=0 EL
     SEMO(3)=1
1040 IFMO(3)=0 THEN E=1 ELSE E=0
1050 IFMO(3)=0 THEN GOSUB1090 ELSE1070
1060 RETURN
1070 E=0:IFMO(4)=0 THEN GOSUB1090 ELSE RETURN
1080 GOTO560
1090 FORI=1 TO 10
1100 GOSUB1120
1110 NEXTI
1120 PRINT@320,D$(E),NO$(E),D$(F),NO$(F)
1130 FORH=1 TO 300:NEXTH
1140 PRINT@320,STRING$(62,32):FORH=1 TO 300:N
    EXTH
1150 RETURN
1160 VIM=1119:CLD:PRINT"OVERALL MILES PER GAL
    LON":PRINT:PRINT"COMMAND LIST # 2"
1170 PRINT:PRINT"1-DISPLAY MILES/GALLON"
1180 PRINT"2-RETURN TO COMMAND LIST #1"
1190 INPUT"ENTER COMMAND BY NUMBER";R:IFR<1
    OR R>2 THEN 1160
1200 ON R GOSUB 1210,1400:GOTO1160
1210 X=0:Y=0:Z=0:FORJ=4 TO N
1220 X=(O(N)-O(4)):Y=Y+G(J)
1230 NEXTJ
1240 Z=X/Y:Z=INT(Z*100):Z=Z/100
1250 CLS:PRINT"TOTAL MILEAGE",X:PRINT"TOTAL
    GALLONS";Y:PRINT"MILES/GALLON",Z
1260 IFG(3)-Z=>1 THEN GOSUB 1300
1270 IFZ>G(3) THEN D$(3)=D$(N) ELSE D$(3)=D$(3)

1280 IFZ>G(3) THEN G(3)=Z ELSE G(3)=G(3)
1290 GOTO1380
1300 FORI=1 TO 10
1310 GOSUB 1330
1320 NEXTI
1330 PRINT@320,D$(N),"POOR PERFORMANCE"
1340 FORH=1 TO 300:NEXTH
1350 PRINT@320,STRING$(31,32):FORH=1 TO 300:N
    EXTH
1360 RETURN
1370 RETURN
1380 GOSUB1440
1390 RETURN
1400 GOTO70
1410 VIM=1055:CLS:PRINT"COST/MILE =";"$";AE/X
1420 GOSUB1440
1430 RETURN
1440 PRINT@448,"PRESS ENTER TO RETURN"
1450 B$="":R$=INKEY$:IFR$=B$ THEN 1450
1460 IFPEEK(342)=247 THEN 1540 ELSE 1470
1470 RETURN
1480 POKE153,10:POKE154,66:POKE115,80
1485 PRINT#-2,"N","ACCNT","DATE","MIL.","FUEL",
    "NOTE","AMOUNT"
1490 PRINT#-2,CHR$(10)
1500 FORJ=1 TO N
1510 PRINT#-2,J,A(J),D$(J),O(J),G(J),NO$(J),C(J
    )
1520 NEXTJ
1530 RETURN
1540 ZW=0:FORZX=1024 TO VIM:ZW=ZW+1
1550 ZY=PEEK(ZX)
1560 IFZY=>96 AND ZY<128 THEN ZY=ZY-64 ELSE ZY=
    ZY
1570 PRINT#-2,CHR$(ZY);
1580 IFZW=32 THEN PRINT#-2,STRING$(48,32);
1590 IFZW=>32 THEN ZW=0
1600 NEXTZX:PRINT#-2,CHR$(32)
1610 RETURN
```

©

Joysticks And Sprites On The Commodore 64

Sheldon Leemon

As the owner of an Atari 800 computer, I welcomed Commodore's announcement of the Model 64 computer, because it closely parallels the Atari in its consumer orientation. One example is the inclusion of two ports for Atari-type joystick controllers. These controllers provide a simple way for the user to interact with any type of program, including, of course, arcade games.

A Fascinating Chip

When I bought the computer, however, I discovered, to my dismay, that the consumer-oriented design approach did not seem to carry through to the BASIC interpreter and *User's Guide*. Not only was there no BASIC command for reading the joystick controllers, but the BASIC manual also made no mention whatever of these ports! This meant that if I discovered how to use these sticks any time soon, I would have to play hardware detective.

Fortunately, the 64 is quite similar to the VIC-20 in a number of ways. Since the VIC reads the joystick through the VIA (Versatile Interface Adapter) chip, it stands to reason that the 64 would read its joystick through the analogous CIA (Complex Interface Adapter) chip. An early memory map from Commodore shows CIA #1 to be addressed at location DC00, or 56320 decimal. The CIA is a fascinating I/O chip, and could well serve as the basis for an article in itself, but here I'll focus attention on the registers that read the joysticks.

Like the VIC-20, the 64 uses Peripheral Data Registers A and B to read these sticks, and I/O (input/output) through these registers is controlled by Data Direction Registers A and B. These registers are addressed at the chip's first four locations, so that on the 64 Data Register A is addressed at 65320, Register B is addressed at 56321, and Data Direction Registers A and B are addressed at 56322 and 56323, respectively.

Reading The Joysticks

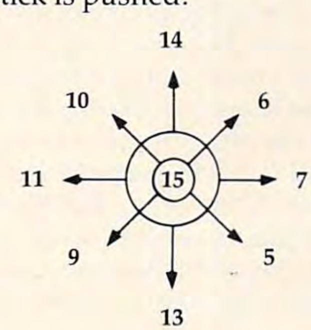
Knowing this, with a bit of trial and error I was able to figure out how to read the joysticks. A quick try seemed to indicate that it was not necessary to write to the Data Direction Registers before reading the sticks, as must be done on the VIC-20. Checking the values of Registers A and B while moving joysticks connected to Control Ports 1 and 2 revealed that the data from the stick connected to Control Port 1 appeared in Register B, and that the data from the stick in Port 2 showed up in Register A. This observation conflicts slightly with the memory map which Jim Butterfield published in the October issue of **COMPUTE!**. That map shows that Register A controls Joystick 0, and Register B controls Joystick 1.

The relationship of the data returned in the register to the direction of stick movement is exactly the same as on the Atari. Each of the low bits (0-3) corresponds to one of the switches that is closed by moving the stick in one of the four primary directions. These bits are normally set to 1, but are reset to 0 when the corresponding switch is closed. Bit 0 corresponds to the up switch, bit 1 corresponds to the down switch, bit 2 is left, and bit 3 right. Bit 4 is used to read the joystick trigger button. It is set to 1 normally, and reset to 0 if the button is pushed.

What this means to the hardware-weary reader who has borne with me thus far, patiently waiting for an explanation in plain English of how to use the Commodore 64 joysticks, is that it takes only a couple of BASIC statements to do the job. Those familiar with the Atari system of numbering the joystick positions (as I am) may want to use the following statements:

```
S1=PEEK(56321) AND 15: REM Reads Stick 1
S2=PEEK(56320) AND 15: REM Reads Stick 2
```

Because these registers can contain irrelevant information in bits 4-7, the logical AND is used to mask (block out) those bits. The figure below shows the way in which the number returned in variable S1 or S2 corresponds to the direction in which the stick is pushed.



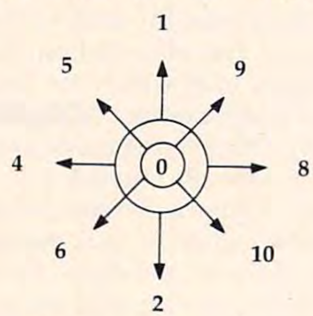
To read the trigger buttons, the following statements will return a 1 if a button is pressed, and a 0 if it is not:

```
T1=-((PEEK(56321) AND 16)=0)
T2=-((PEEK(56320) AND 16)=0)
```

Of course, if you prefer a system where the variable will be 0 when the stick is not pressed, you can use the logical operator NOT to adjust the values accordingly.

```
S1=NOT PEEK(56321) AND 15
S2=NOT PEEK(56320) AND 15
```

This will produce the following pattern:



A Keyboard Bonus

The variations on these basic schemes are limited only by your applications. If you are using the joystick for an action game, for example, you may want to read the change in horizontal position and vertical position separately. You can do this with the following formulas:

```
H1=((PEEK(56321) AND 15)=4) - ((PEEK(56321)
   AND 15)=8)
H2=((PEEK(56320) AND 15)=4)-((PEEK(56320 AND
   15)=8)
V1=((PEEK(56321) AND 15)=1) - ((PEEK(56321) AND
   15)=2)
V2=((PEEK(56320) AND 15)=1)-((PEEK(56320) AND
   15)=2)
```

The value of H1 will be 1 if the stick is pressed to the right, -1 if the stick is pressed to the left, and 0 if centered. Likewise, the value of V1 will be -1 for an upward press, 1 for a downward press, and 0 if the stick is centered. If you wish, you can even read each switch separately. Program 1, short and not exciting, demonstrates the technique.

One interesting sidelight demonstrated with this program is the fact that some CIA registers that are used to read the joysticks are used also to read the keyboard. The four keys at the top left of the keyboard (Control, Left Arrow, 1, and 2) are read exactly the same as joystick switches 0-3. While you are running Program 1, try pressing these keys, and you will see what I mean.

Pressing the Control key has the same effect as moving the stick to the left, while the Left Arrow, 1, and 2 keys function like a joystick moved down, up, and to the right, respectively.

Graphics Movement

The initialization routine, which I have put out of the way at the back of the program, starting with line 1000, sets up a flying saucer in double width, and then returns to the movement loop at line 2. The ON-GOSUB routes the program to the proper line number without having to test each stick position, which would slow down the loop.

There are a couple of points to note. First, the registers that designate sprite horizontal and vertical positions are not write-only registers, as are the Atari horizontal position registers. This means that you can find out the current position of the sprite just by reading those registers, without having to set up separate RAM variables to keep track of them as must be done on the Atari. I set up variables %X and %Y in Program 2 only for purposes of readability.

To move a sprite one position to the right, we need only read the current horizontal position, add 1, and POKE that number back into the horizontal position register. Of course, you must keep in mind that you can't POKE in a value less than 0 or greater than 255. If you examine the move-down and move-up subroutines at lines 80 and 90, you will see that I have incorporated logical statements to move the sprite to the bottom of the screen if it hits the upper limit, and which will move it to the top if the value tries to get below 0. This wraparound feature guarantees that no errors will result from trying to POKE in an illegal quantity.

The Horizontal "Seam"

A more complicated situation arises when we deal with horizontal movement. Because there are 320 horizontal positions available, but only 256 combinations which can be accessed from the horizontal position register, we need to set the Most Significant Bit in the register located at 53264 whenever we wish to use a horizontal position between 256 and 320. Any time the sprite moves into or out of this zone, therefore, special handling of this bit will be required.

Accordingly, the horizontal movement routines (lines 40-45 and 70-75) have to test to see if this "seam" is encountered before moving the sprite. If the horizontal position register reads 0, for example, we don't know whether the sprite is located at the left edge of the screen or at the "seam" (i.e., location 256) until we check the MSB register. This extra checking is time consuming, and as a result the saucer moves noticeably faster up and down than it does right and left.

Because of the slowness of the motion in BASIC, I have multiplied all motion by the factor WUN, which is defined in line 1005, and which can be set from 1 to 3. When its value is 1, the motion is very smooth, but extremely slow. When it is 3, each push of the stick changes the position of the sprite by three places, speeding up the motion, but making it somewhat jerky.

Machine Language Motion

The best solution to the problem of achieving quick, smooth motion is the use of a machine language subroutine which will read a joystick, and move the sprite accordingly. Program 3 uses just such a subroutine. Though I POKE it into memory starting at C000 (49152 decimal), it is completely relocatable.

If it later proves that this large block of free RAM can be better used otherwise, you will be able to move the routine with no rewriting. You should be aware, however, that, as written, the routine checks only the joystick in Port 1, and moves only Sprite 0 in response to movement of that stick. Since some lines of Program 3 duplicate those of Program 2, you may want to edit the latter program rather than typing in Program 3 from scratch.

One difference that you will notice immediately is that this program asks you to select a speed (you should respond with a value from 1-5). The reason for this is that I wanted to demonstrate the degree to which even a machine-language subroutine is slowed down by BASIC. At Speed 1, each time through the loop the program calls the subroutine once and returns to BASIC. Though this produces smooth motion, it is still somewhat slow. At Speed 2, the program calls the subroutine twice in a row before returning, and so on up to Speed 4, which produces rather quick motion. At Speed 5, the machine language subroutine goes into a continuous loop, without ever returning to BASIC. At this speed, if you push on the stick diagonally, it will appear as if there are dozens of saucers on the screen at once!

Though my examples may seem most applicable to game programs, do not overlook the joysticks as input devices for more "mundane" tasks. Because each stick has only four switches, it limits the number of choices available to the user. It therefore reduces the number of mistakes that can be made, as compared with a keyboard, which has over 60 keys, each key having both a shifted and non-shifted value.

Program 1.

```
10 FOR I=1 TO 25:DOWN$=DOWN$+CHR$(17):NEXT:HO
    ME$=CHR$(19):PRINTCHR$(147);CHR$(5)
15 PRINT" THIS PROGRAM READS STICK #1":PRINT"
     INSERT JOYSTICK, AND MOVE IT AROUND!
    "
20 S=NOT PEEK(56321) AND 15
30 UP=S AND 1:IF UP THEN PRINT HOME$;LEFT$(DO
    WN$,10);TAB(15);"UP   ";:GOTO 50
40 DOWN=S AND 2:IF DOWN THEN PRINT HOME$;LEFT
    $(DOWN$,10);TAB(15);"DOWN ";
50 LEFT=S AND 4:IF LEFT THEN PRINT HOME$;LEFT
    $(DOWN$,10);TAB(25);"LEFT ";:GOTO70
60 RIGHT=S AND 8:IF RIGHT THEN PRINT HOME$;LE
    FT$(DOWN$,10);TAB(25);"RIGHT";
70 IF S=0 THEN PRINT HOME$;LEFT$(DOWN$,10);TA
    B(15);"                    "
80 GOTO 20
```

Program 2.

```
1 GOTO 1000
2 S=PEEK(S0)AND15:ONSGOSUB3,3,3,3,20,30,40,3
    ,50,60,70,2,80,90,3:GOTO2
3 RETURN
20 GOSUB 40:GOSUB 80:RETURN
30 GOSUB 40:GOSUB 90:RETURN
40 X%=X%+WUN :IF X%>255 THEN X%=0:POKE SP+16,
    1
43 IF X%>65 AND PEEK(SP+16)=1 THEN POKE SP+16
    ,0:X%=0
45 POKEHP,X%:RETURN
50 GOSUB 80:GOSUB 70:RETURN
60 GOSUB 90:GOSUB 70:RETURN
70 X%=X%-WUN:IF X%<1 AND PEEK(SP+16)=1 THEN X
    %=255:POKE SP+16,0
73 IF X%< 1 AND PEEK(SP+16)=0 THEN X%=65:POKE
     SP+16,1
75 POKEHP,X%:RETURN
80 Y%=Y%+WUN+HI * (Y%>HI):POKEVP,Y%:RETURN
90 Y%=Y%-WUN-HI * (Y%<WUN):POKEVP,Y%:RETURN
1000 FORI=871TO895:POKEI,0:NEXT:FOR I=832TO870:
    READA:POKEI,A:NEXT:SP=53248
1005 HP=SP:VP=SP+1:X%=160:Y%=100:WUN=3:HI=252:S
    0=56321
1010 POKESP+21,1:POKE2040,13:POKESP+39,6:POKESP
    +29,1:POKEHP,X%:POKEVP,Y%
1020 POKESP+32,0:POKESP+33,0:PRINTCHR$(147)
1030 FORI=1 TO 50:POKE 1024+INT(RND(0)*1000),46
    :NEXT
1040 DATA 0,56,0,0,124,0,0,254,0,0,170,0,1,171,
    0,15,255,224,15,255,224,13,85,96
1050 DATA 13,85,96,15,255,224,15,255,224,0,254,
    0,0,124,0
1060 GOTO 2
```

Program 3.

```
10 PRINTCHR$(147);CHR$(5): INPUT"SPEED ";S:GO
    TO 1000
20 ON S GOTO 30,40,50,60,70
30 SYS(49409):GOTO 30
40 SYS(49406):GOTO 40
50 SYS(49403):GOTO 50
60 SYS(49400):GOTO 60
70 SYS(49413):GOTO 70
1000 FORI=871TO895:POKEI,0:NEXT:FOR I=832TO870:
    READA:POKEI,A:NEXT:SP=53248
1010 POKESP+21,1:POKE2040,13:POKESP+39,6:POKESP
    +29,1:POKESP,160:POKESP+1,100
1020 POKESP+32,0:POKESP+33,0:PRINT CHR$(147)
1030 FORI=1 TO 50: POKE 1024+INT(RND(0)*1000),4
    6:NEXT
1040 DATA 0,56,0,0,124,0,0,254,0,0,170,0,1,171,
    0,15,255,224,15,255,224,13,85,96
1045 DATA 13,85,96,15,255,224,15,255,224,0,254,
    0,0,124,0
1050 FOR I=1 TO 101:READ A:POKE 49151+I,A:NEXT
1055 FOR I=1 TO 19:READ A:POKE 49399+I,A:NEXT:G
    OTO 20
1060 DATA 173,1,220,74,176,3,206,1,208,74,176,3
    ,238,1,208,74,176,38,173
1070 DATA 0,208,208,15,173,16,208,41,1,240,12,1
    73,16,208,41,254,141,16
1080 DATA 208,206,0,208,96,173,16,208,9,1,162,6
    3,141,16,208,142,0,208,96
1090 DATA 74,176,32,238,0,208,240,28,173,16,208
    ,41,1,240,20,169,64,205
1100 DATA 0,208,208,13,173,16,208,41,254,162,0,
    141,16,208,142,0,208,96
1110 DATA 173,16,208,9,1,141,16,208,96
1200 DATA 32,0,192,32,0,192,32,0,192,32,0,192,9
    6,32,0,192,76,5,193
```

©

Assembly Language And The PET

R. D. Wink

Designed for those as yet unfamiliar with machine language programming, this tutorial presents a detailed analysis of a simple machine language program which computes factorials.

Interested in machine language programming? Find the books tough going? This article could be for you!

As a PET owner who is fairly competent in BASIC, I have often wanted to try my hand at writing machine language programs. Yet, as I worked through several texts on the topic, I found that they are apparently written for readers who already have a good grasp of the basics. Perhaps I'm a little slow at catching on, so this article is written for those who, like me, might be interested in a line-by-line analysis of a simple program. The program calculates the factorial function to a precision in excess of 80 digits and is written in 6502 assembly language. Hex dumps and a BASIC loader program are provided for those who do not have access to an assembler.

The factorial is a mathematical function useful in probability studies. N factorial (written N!) is defined as:

N! = N x (N-1) x (N-2) x...2 x 1

As an example, 5! = 5x4x3x2x1 = 120. A few moments thought will show that as N gets bigger, the value of N! rapidly becomes vast. Indeed 69! is of the order of ten to the 99th. This function was chosen because it is complicated enough to require multiple-precision (see below), yet it avoids the problems that decimal fractions cause the machinne language programmer.

Since the 6502 microprocessor does not have a built-in multiplication function, multiplication must be accomplished by repeated addition. Also, since the 6502 is an eight-bit processor, the largest number it can handle in one operation is 255 – not a very promising start for a number like 69!. In fact, using only a single eight-bit word (byte), the largest factorial that can be computed is 5! or 120. 6! is 720, and this is too large to be held in a single byte. Obviously, it will be necessary to use a series of consecutive bytes to represent the big numbers involved and then to handle these numbers a byte at a time. This is what is meant by the term "multiple precision."

We shall first write a BASIC program which computes factorials (though only to nine-digit precision), and then we shall make a line-by-line comparison between this and the assembly code version. The first version in Program 1 uses multiplication. Line 5 sets the initial value of the product P to one. Line 10 calls the required factorial (e.g., 3! means that N is 3). Lines 15 to 25 multiply the existing product value by values of N which are reduced by one each time around the loop. The first time through line 15, P is three. The next time P is 3x2 or six, and the last time P is 3x2x1 which is still six.

We shall now replace the multiplication in line 15 with a subroutine which does the same job, but by using repeated addition (Program 2). This subroutine requires the use of two new variables C and M. C is a counter which is set equal to N at the start of the subroutine. It is used to count the number of additions which have taken place. M is a variable which holds the successive sums needed in the multiplication process.

For example, suppose the routine is to do the multiplication 7x6 where P = 7 and N = 6. Lines 110-120 cause the number seven to be added to the variable M six times:

7 + 7 + 7 + 7 + 7 + 7 = 6 x 7

The result, 42, is stored in the variable P prior to a return to the main program. The reader should understand that the product (line 15) P = P*N has been replaced by an equivalent subroutine which uses only addition.

As we discuss the assembly language version of the program, we shall make frequent reference to three registers in the microprocessor. The contents of any byte of memory may be copied into the accumulator, the X or the Y register, with the


JINSAM™
EXECUTIVE™
space
age
micro
software
OFFICIALLY APPROVED BY COMMODORE
Used at NASA,
Kennedy Space Center
With Multiple Applications Related
to the Columbia Space Shuttle Project
including rescue operations, statistical
reports, inventory and vehicle tracking.
JINSAM EXECUTIVE™
has broken the 10,000 record limit. You may now have up to 65,000 records in one database.
We also have included a free form report generator for data entry, eliminating the need for WordPro™ and have included automatic mathematical relations eliminating the need for VisiCalc™. However, you still have these superb interfaces available.
Executive™ will be available for CBM and IBM personal computers.
JINI MICRO-SYSTEMS, Inc.
DATABASE MANAGEMENT SYSTEM DESIGN
BOX 274 KINGSBRIDGE STN., RIVERDALE, N.Y. 10463 (212) 796-6200

appropriate load instruction, LDA, LDX, or LDY. The contents of these registers may be sent to any memory location using the store instructions STA, STX, and STY. Moving a number between these registers is managed with the transfer instructions (e.g., TXA moves the contents of the X register into the accumulator). Arithmetic is done using the accumulator. The contents of a memory location may be added to the contents of a memory location may be added to the contents of the accumulator using the ADC or "add with carry" instruction. The result of the addition remains in the accumulator, and, if the sum exceeds 255, a special bit called the carry flag is set so that on the next addition the "carry one" is added in.

We are now ready to translate the BASIC program (Program 2) into a single precision assembly language program. A corresponding BASIC method is in parentheses in the comment column.

1. * = $33A	An assembler instruction that sets the start of memory into which the machine code is to be assembled. Hex 33A is the start of the second cassette buffer, which is usually not used and is a "safe" location.
2. SED	Puts the 6502 into decimal mode. Easier for us beginners to work with than hex. Each byte contains a two-digit decimal number less than 100.
3. JSR INITP	Jump to subroutine "INITP" which initializes P (5 P = 1).
4. FOUR LDA FACT	Put the value of the required factorial into the accumulator (10 INPUT N).
5. STA C	Store the contents of the accumulator in the variable C (100 C = N).
6. JSR ZERM	Jump to subroutine ZERM which initializes the variable/memory location M (105 M = 0).
7. THREE JSR ADD	Jump to subroutine ADD (110 M = M + P).
8. DEC C	Decrement the value of C (115 C = C-1).
9. BNE THREE	If the last value operated on (C) is not zero, then branch to label THREE, line 7 above (120 IF C‹›9 THEN 110).
10. JSR MTOP	Jump to subroutine M TO P (125 P = M).
11. DEC FACT	Decrement the value of FACE (20 N = N-1).
12. BNE FOUR	Branch to label FOUR if the last operand "FACT" is not zero (25 IF N‹›0 THEN 15).
13. CLD	Clear decimal mode. If we don't return microprocessor to its normal hex mode, PET throws a fit on return to BASIC.
14. BRK	(35 STOP)

Now the four subroutines referenced above:

15. INITP LDA #1	Place 1 in the accumulator.
16. STA P	Place 1 in P.
17. RTS	Return.
18. ZERM LDA #0	Place zero in the accumulator.
19. STA M	Place zero in M.
20. RTS	Return.
21. ADD CLC	Clear the "carry" flag prior to addition.
22. LDA M	Load the accumulator with contents of M.
23. ADC P	Add to the contents of the accumulator the contents of P.
24. STA M	Store the result of the addition in M.
25. RTS	Return.
26. MTOP LDA M	Copy the value of M into the accumulator.
27. STA P	Store the value in location P.
28. RTS	Return.
29. FACT .BYTE 4	Assembler instructions which reserve space for the variables.
30. C .BYTE 0	FACT, C, P, and M. (The precise method of doing this varies depending on the assembler used.)
31. P = *	
32. M = * + 1	
33. .END	

The program listed above can be assembled and run, though it is probably not worth the trouble of typing it all in and assembling it, merely to have the number 4! or 24 appear in the byte which P represents. Program 3 contains the assembled code, and the reader may wish to use the resident monitor in the Upgrade ROM PET to test the program. Type SYS 4 and press RETURN in order to call the monitor. Display the appropriate memory locations by typing M 033A, 037A RETURN. The screen should fill with hex codes, which should be carefully replaced by those listed in Program 3. At the end of each line, be sure to press RETURN in order to enter the code into memory.

After 037A has been completed, type G 033A, RETURN in order to run the program from the start. After a moment, the microprocessor registers should be displayed and the reader will note that the accumulator (AC) contains the number 4! or 24 where 4 was the number placed in the variable FACT. The memory locations can be relisted by moving the cursor back up to the line M 033A, 037A and pressing RETURN. Watch the location 0379, which is P, change to 24. The location 0377, immediately after hex code 60, is FACT. Use the cursor controls to change it from 00 to 05. Press RETURN, cursor down to G 033A and press RETURN, again. Relist the memory locations 033A, 037A and note that P (0379) now contains the number 20, which is the last two digits of 5! or 120. Code X will return control to BASIC.

It is fairly simple to compare the assembled code in Program 3 with the assembly language program. Looking at the line.: 033A F8 20 59 03 AD 77 03, we may interpret the codes as F8 = SED, 20 = JSR, 59 03 is address 359 where subroutine INITP starts, AD = LDA, and 77 03 refers to location 377, which is FACT.

This article has so far described a simple assembly language program which computes factorials. Since the routine is only single precision, the largest factorial that can be handled is 5! or 120. The reader will have noticed that subroutines were extensively used; although this slows down the execution time, it will now make program revision much simpler.

The rest of the article describes the modifications necessary to incorporate multiple precision

and a way of linking the machine code to a simple BASIC program which displays the results in a more palatable form.

We shall extend the precision to 40 bytes (or 80 significant decimal digits) merely by altering the subroutines. The mainline program is unchanged. Eighty-digit precision was chosen as being the maximum that could be fitted into the memory available in the second cassette buffer.

The variables, P and M, should now be considered as 40 consecutive bytes each, rather than as the single byte allowed in the previous subroutines. Each of the bytes in P can be addressed as P+0, P+1, P+2...P+39, and similarly with M. In order to avoid massive duplication of code, a technique called "indexed addressing" will be used.

Consider the subroutine INITP, which originally had the function of initializing P to a value of one. In the multiple precision scheme, we shall need to place 00 in locationns P+0, P+1...P+38 inclusive, and 01 in the least significant byte P+39. In this way P is initialized to 000000000000 00000000000000000000000000000000000000000 00000000000000000000000001.

A loop is used to place the zeros in the first 39 bytes, together with the instruction STA P,X. This instruction copies the contents of the accumulator into location P+X of the variable P, where X is the contents of the X Index register. If X is initialized to 38, and the accumulator contains zero, then the instruction STA P,X will place zero in the location P+38. The value of X can then be decremented (reduced by one) each time around the loop so that P+37, P+36, etc., will each have zero copied into them.

```
INITP  LDX #39     Place 39 into the X index register.
       LDA #0      Place zero into the accumulator.
ONE    DEX         Decrement the X register (first value=38).
       BMI FIN     If the X register contents is less than zero,
                   branch to label FIN.
       STA P,X     Store zero in byte P+X.
       JMP ONE     Go to label ONE.
FIN    LDA #1      Place one in the accumulator.
       STA P+39    Store one in the byte P+39.
       RTS         Return.
```

Subroutine ZERM sets the value of M equal to zero and so must be modified to place 00 in each of the bytes M+0, M+1, M+2...M+39.

```
ZERM   LDY #40     Place 40 into the Y index register.
       LDA #0      Place zero into the accumulator.
TWO    DEY         Decrement the Y register (first value=39).
       BMI RET     Branch on a negative value of Y to label RET.
       STA M,Y     Store zero in byte M+Y.
       JMP TWO     Go to label TWO.
RET    RTS         Return.
```

Subroutine ADD first loads the accumulator with the least significant byte of M (M+39). The least significant byte of P (P+39) is added to it, and the carry flag is set if the result exceeds 99. The sum is then placed in byte M+39, replacing the previous contents. The process is then repeated with bytes M+38 and P+38, except that the "carry" will be added in if appropriate. The addition of corresponding bytes continues for P+37, P+36, etc., all the way down to P+0.

```
ADD    LDY #40     Place 40 in the Y register.
       CLC         Clear the carry flag prior to addition.
LOOP   DEY         Decrement the Y index register (first
                   value=39).
       BMI DONE    Branch on a negative value of Y to label DONE.
       LDA M,Y     Place the contents of byte M+Y in the
                   accumulator.
       ADC P,Y     Add (with carry) the contents of P+Y to
                   accumulator.
       STA M,Y     Place the sum in byte M+Y.
       JMP LOOP    Go to label LOOP.
DONE   RTS         Return.
```

Subroutine MTOP transfers the contents of all 40 bytes of M to the corresponding 40 bytes of P.

```
MTOP   LDX #40     Place 40 in the X register.
SIX    DEX         Decrement the X index register (first
                   value=39).
       BMI FIVE    Branch on a negative value of X to label FIVE.
       LDA M,X     Place byte M+X in the accumulator.
       STA P,X     Copy the contents of the accumulator into
                   byte P+X.
       JMP SIX     Go to label SIX.
FIVE   RTS         Return.
```

These subroutines, together with the original mainline program, can be assembled and run using the resident machine language monitor. An assembled version is listed in Program 4, and it can be entered using the monitor. Location 039C contains FACT and this is set to a value of six in Program 4. The 40 bytes from 039E to 03C5 are the variable P. If the program is run with the instruction G 033A and memory listed using M 039E, 03C5, then the answer 720 can be seen in the least significant bytes.

The required factorial should be converted to hex and placed in location 039C prior to running the program. If 20! is to be computed, then hex value 14 (16+4) is placed in this byte. The answer will appear in decimal form when M 039E, 03C5 is listed. For 20!, the result is 2432902008176640000 and the largest factorial which can be displayed fully is 58! (hex 3A). If it is required to link the program to BASIC, the BRK instruction (00) must be replaced by RTS (60) in location 0358. You can then save the routine using the monitor and write a BASIC program which calls it using SYS 826.

Program 5 is a BASIC listing which obtains the machine codes from data statements and POKEs them into the second cassette buffer. Note that each hex code has been converted into its decimal equivalent as required by the POKE instruction. The second portion of the program requests the required factorial and prints the answer in two lines on the screen.

Should greater precision be required, the

code will have to be assembled into the high end of RAM memory, so that a BASIC calling program can be used. The end of BASIC pointer can be lowered and the precision extended up to 255 bytes or 510 digits. I hope that readers will find this article a simple way of getting their feet wet exploring 6502 machine and assembly language.

Program 1.

```
5  P=1
10 INPUT N
15 P=P*N
20 N=N-1
25 IF N<>0 THEN 15
30 PRINT P
35 END
```

Program 2.

```
5  P=1
10 INPUT N
15 GOSUB 100
20 N=N-1
25 IF N<>0 THEN 15
30 PRINT P
35 STOP
100 C=N
105 M=0
110 M=M+P
115 C=C-1
120 IF C<>0 THEN 110
125 P=M
130 RETURN
135 END
```

Program 3.

```
033A F8 20 59 03 AD 77 03 8D
0342 78 03 20 5F 03 20 65 03
034A CE 78 03 D0 F8 20 70 03
0352 CE 77 03 D0 E7 D8 00 A9
035A 01 8D 79 03 60 A9 00 8D
0362 7A 03 60 18 AD 7A 03 6D
036A 79 03 8D 7A 03 60 AD 7A
0372 03 8D 79 03 60 04 00 00
037A 00 00 00 00 00 00 00 00
```

Program 4.

```
033A F8 20 59 03 AD 9C 03 8D
0342 9D 03 20 6C 03 20 7A 03
034A CE 9D 03 D0 F8 20 8D 03
0352 CE 9C 03 D0 E7 D8 00 A2
035A 27 A9 00 CA 30 06 9D 9E
0362 03 4C 5D 03 A9 01 8D C5
036A 03 60 A0 28 A9 00 88 30
0372 06 99 C6 03 4C 70 03 60
037A A0 28 18 88 30 0C B9 C6
0382 03 79 9E 03 99 C6 03 4C
038A 7D 03 60 A2 28 CA 30 09
0392 BD C6 03 9D 9E 03 4C 8F
039A 03 60 06 00 00 00 00 00
```

Program 5.

```
10 DATA 248 , 32 , 89 , 3 , 173 , 156 , 3 , 141
15 DATA 157 , 3 , 32 , 108 , 3 , 32 , 122 , 3
20 DATA 206 , 157 , 3 , 208 , 248 , 32, 141 , 3
25 DATA 206 , 156 , 3 , 208 , 231 , 216 , 96 , 162
30 DATA 39 , 169 , 0 , 202 , 48 , 6 , 157 , 158
35 DATA 3 , 76 , 93 , 3 , 169 , 1 , 141 , 197
40 DATA 3 , 96 , 160 , 40 , 169 , 0 , 136 , 48
45 DATA 6 , 153 , 198 , 3 , 76 , 112 , 3 , 96
50 DATA 160 , 40 , 24 , 136 , 48 , 12 , 185 , 198
55 DATA 3 , 121 , 158 , 3 , 153 , 198 , 3 , 76
60 DATA 125 , 3 , 96 , 162 , 40 , 202 , 48 , 9
65 DATA 189 , 198 , 3 , 157 , 158 , 3 , 76 , 143
70 DATA 3 , 96
75 FOR K = 826 TO 923
80 READ A:POKE K,A
85 NEXTK
90 :
95 REM BASIC CALLING PROGRAMME.
99 :
100 INPUT"FACTORIAL";N
105 POKE 924,N
110 SYS826
115 FOR K = 0 TO 39
120 A=PEEK(926+K)
125 H=INT(A/16)
130 L=A-H*16
135 A=10*H+L
140 IF A<10 THEN A$="0"+RIGHT$(STR$(A),1):GOTO 150
145 A$=RIGHT$(STR$(A),2)
150 S$=S$+A$
155 NEXT K
160 PRINT S$:S$=""
165 GOTO 100
170 END
```

©

Programming The PET/CBM

by Raeto Collin West

The book described by Jim Butterfield as

"...unquestionably the most comprehensive and accurate reference I have seen to date..."

The Reference Encyclopedia for Commodore 2000, 3000, 4000, and 8000 series computers and peripherals.

Here's just a sample of reviewer and reader reaction:

From reviewers:

Educational Computing Review by Stephen Potts
"Of all the books I have read on the PET this book *Programming the PET/CBM* by Raeto West must rank as one of the most comprehensive and readable accounts on the PET that I have ever had the pleasure to see...

"If you wish to get more from your PET than arcade games and simple teaching programs then this book is a must for your bookshelf. It does not matter whether you run on BASIC 1, BASIC 2, or BASIC 4 since all routines are supplied with addresses and changes to make them run on any machines wherever possible...

"...this book, with its lucid explanations of the PET, its useful routines and programming hints, is an essential purchase."

IPUG Magazine Review (British PET User Group) by Ron Geere
"This publication represents over a year's intensive research ... and the resulting product is a valuable work of reference. A tremendous amount of useful information has been packed in this 500+ page work at which I was so over-awed that I did not know how to start this review at first...

"This book is a must for every CBM/PET user."

From readers:

"...a book the average to advanced user cannot afford *not* to possess..."

"My copy of your '*Programming the PET/CBM* has been in daily use for nearly a month and I am finding it totally addictive, suffering severe withdrawal symptoms whenever I try half-heartedly to move on to other reading matter. It is without doubt the best book on its subject available today..."

"I have recently acquired a copy of your book *Programming the PET/CBM* and must congratulate you on its concept and on packing in so much detail. It's so very much better than anything I have had up to now that it'll be my constant reference manual."

"I have received my copy of *Programming the PET/CBM* by Raeto West and I have recommended it to several of my students. This book is so valuable that I cannot now afford to be without it."

Published exclusively in North America by **COMPUTE! Books**. The book is an astonishing reference manual of useful information. Contents include this and much more:

1 Introduction and overview: Plan of the book, sources of information, features and chronology of CBM hardware.
2 BASIC and how it works: Storage of BASIC and its variables; tokens, pointers, syntax; optimising BASIC.
3 Program and system design: Capabilities of the equipment; charts, algorithms, space, timing.
4 Effective programming in BASIC: Seventeen examples, including subroutines, dates, DATA, INPUT, rounding.
5 Alphabetic reference to BASIC keywords: Full descriptions, with examples, of all keywords, with methods for adding additional commands not present in CBM BASIC, e.g. AUTO, DEL, OLD, POP, PRINT USING, SORT, VARPTR.
6 Disk drives: Descriptions of operation and workings of disk drives, with BASIC and machine-code examples; bugs.
7 Alphabetic reference to disk BASIC commands: BASIC 4 disk commands with examples and notes.
8 Other peripherals and hardware: Tape storage and handling; printers; modem; keyboard; EPROMs; reset switches.
9 Graphics and sound: Tables of CBM characters; CRT chip; animation, bar plots, 80 by 50 etc.; user-port sound.
10 The transition to machine-code: Introductory concepts; a BASIC monitor; use of MLM, Supermon, Extramon; easy examples.
11 More 6502 machine-code: 6502 hardware features; eighteen common problems in programming; debugging.
12 Alphabetic reference to 6502 opcodes: Examples, notes, and explanations on each opcode from ADC to TYA.
13 Using ROM routines: IRQ, NMI, RESET; the Kernel; examples – modifying LIST; ordinary and relocating loaders.
14 Effective 6502 programming: Assemblers; CHRGET and wedges; PIAs, VIA, IEEE; common mistakes.
15 Index to BASIC ROMs and RAM: Memory map of RAM and ROM, detailing and comparing BASICs 1, 2, and 4.
16 Mathematical programming: Precision; equations; statistics; simulation; finance; calculus; machine-code.
17 Programming in business and education: Examples, applications and pitfalls in business and education.
Appendices: 6502 reference charts; Supermon listings; ASCII; glossary

Plus many programs, diagrams and charts. Paperback, 504 pages. ISBN 0 942386 04 3. **$24.95.**

To Order

Programming The PET/CBM

Call

TOLL FREE 800-334-0868
In NC Call 919-275-9809

Or send coupon to

COMPUTE! Books, P.O. Box 5406, Greensboro, NC 27403

In England, order from Level Limited, P.O. Box 438, Hampstead, London, NW3 1BH. Price in England is £14.90, including P & P.

Please send ________ copy (copies) of **Programming The PET/CBM** at $24.95 each. (In the US and Canada, add $3 shipping and handling. Outside North America add $10 for air mail delivery, $3 for surface delivery).

All orders must be prepaid in US funds (money order, check, or charge).

☐ Payment Enclosed
Please charge my ☐ VISA ☐ MasterCard ☐ Am. Express

Account No. ________ Expires ___/___

Name ________

Address ________

City ________ State ________ Zip ________

Country ________

Allow 4-6 weeks for delivery. Foreign surface delivery allow 2-4 months.

Simple OSI Graphics

Donald Pitts

OSI beginners can get started with graphics and games by following the suggestions in this tutorial on memory mapped graphics for the C4P and C1P.

The new OSI owner need not shy away from graphics simply because there are no special commands for its use.

Many beginning programmers might not consider using graphics, because of the large number of PEEKs and POKEs required. My object here is to help you visualize the way in which graphics works so that the initial barrier will be broken, enabling you to better utilize the potential of your computer.

First, look at the CRT and envision it as separated into little boxes similar to those on graph paper. In each box you may put one symbol from a table of 255 symbols, ranging from numbers and letters to cars and airplanes. Suppose that a symbol is placed in a box that previously contained a different symbol. The new symbol appears, and all traces of the other symbol are lost. You may erase a symbol in a box by putting a blank symbol there.

Now that you have a basic understanding of the concepts involved, we can begin to discuss the actual commands that can be used in graphics. This is where the POKE command comes into play. The POKE command is used essentially as a statement that says "Put this symbol in that box." The POKE command generally takes this form:

POKE /address/ , /ASCII number of character/

The address is usually a number from 0 to 65,535 that indicates a specific place in memory. The OSI screen is *memory mapped*, meaning that the screen display is a representation of the contents of a certain area in memory.

The way in which the memory is interpreted is straightforward. The first byte of the screen memory is shown on the upper left corner of the screen. The consecutive bytes move their way across the upper row from left to right. The byte following the one in the upper right of the screen is represented as the box just immediately below the box in the upper left corner. Thus, the memory is shown in a manner resembling the way you might read a page in a book. (See Figure 1.)

Figure 1.

1	2	3	4	5	6	7	8	9	10	11	12	13	14
15	16	17	18	19	20	21	22	23	24	25	26	27	28
29	30	31											

A theoretical 14-column screen in which the numbers within the boxes represent the byte's position in screen memory

Now for some tangible evidence that what I have been talking about works on your computer. Program 1 fills all of the screen's memory with a single symbol. It moves slowly from one box to another and puts the character into each box. Type it in and RUN it to watch it work.

ASCII And POKE

By now either you already know what an ASCII number is, or you are rather perturbed with me for not defining it for you. It is a standard way of representing characters as numbers. ASCII stands for American Standard Code for Information Interchange. Some quick examples are "A"-65, "1"-49, "P"-80, and "*"-42.

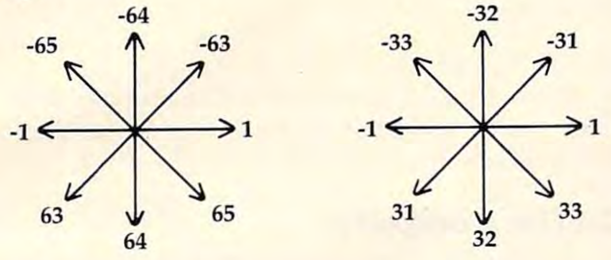
Another statement that is practical in graphics is the PEEK command. This acts similar to a window in which you can see what is displayed in a certain memory location. It has the form:

X=PEEK (/address/)

X could be any variable, and an address such as 54016 could be used. Please note that the parentheses are part of the command and are necessary to avoid a syntax error. After the statement is executed, the variable on the left will contain the ASCII number of the symbol located in the address specified. The realistic uses of PEEK range from checking to see if a tank has been blown up to detecting whether a ball has hit the boundary. Program 2 will first place on the screen the number of men that you specify and then will count how many are on the screen by searching every location in the screen memory for the symbol 240.

Suppose the variable X is assigned to be the location of a car that you have just POKEd on the screen. Now, how do you create the illusion of movement? First we must erase the old car with the statement POKE X,32. The number 32 is the ASCII code for a blank. You must change X to the new location of the car on the screen and then POKE X, 0 since 0 is the number for a car. When you change the car's position, you simply execute the statement X=X+Z, where the variable Z depends upon the direction you wish to move and whether you are using a C1P or a C4P. Here are the values:

Figure 2.



C4P C1P

Thus, if we wanted to move down, the statement would be X=X+64 for a C4P and X=X+32 for a C1P. Program 3 uses these constants from their tables to move a cross in random directions. The program does not check for the edges of the screen to see if the cross has travelled past its boundaries, so if you don't press CTRL-C before it goes far, the program might hang up.

Program 1.

```
10 REM FILL SCREEN WITH AIRPLANE
20 P=PEEK(57088):REM LOOK AT KEYBOARD
30 ST=53315:EN=54205:REM C1P VALUES
40 IFP<129THENST=53376:EN=55295:POKE56832,1:R
   EM C4P VALUES
50 FORLO=STTOEN:POKELO,236:NEXT
60 GOTO 60
```

Program 2.

```
10 REM PUT SPECIFIED NUMBER OF MEN ON SCREEN
15 REM AND COUNT THEM. THERE IS AN INCREASING
    AMOUNT OF ERROR
17 REM AS THE NUMBER OF MEN IS INCREASED DUE
   TO THE FACT THAT
18 REM THE MEN ARE PUT IN THE SAME BOX AS ONE
    ANOTHER.
20 SU=0:ST=53315:EN=54205:X=24:Y=28
30 IFPEEK(57088)<129THENST=53376:EN=55295:X=6
   4:Y=30:POKE56832,1
40 INPUT"NUMBER OF MEN";ME:FORCO=1TO30:PRINT:
   NEXT
50 FORCO=1TOME:POKEST+INT((EN-ST)*RND(1)),240
   :NEXT
60 FORCO=STTOEN:IFPEEK(CO)=240THENSU=SU+1
70 NEXT:PRINT"THERE WERE";SU;"MEN ON THE SCRE
   EN."
```

Program 3.

```
10 REM MOVE CROSS
20 FORX=1TO8:READP(X):NEXT:LO=54016
30 IFPEEK(57088)>128THENFORX=1TO8:READP(X):NE
   XT:LO=53775
40 DATA1,65,64,63,-1,-65,-64,-63
50 DATA1,33,32,31,-1,-33,-32,-31
55 FORX=1TO30:PRINT:NEXT
60 POKELO,219:FORX=1TO30:NEXT:POKELO,32
70 LO=LO+P(INT(RND(1)*8+1)):GOTO60
```

©

COMPUTER CALCULATORS

Jim Butterfield, Associate Editor

Number conversion, masking, even translations of floating point variables are possible when you use the more sophisticated "programmer's calculators." Here are some techniques for using various types of calculators when your computer is doing other things.

Why have a calculator when you already have a computer? Indeed, why would you need a special calculator when the simple four-function units will do all the arithmetic you might need?

The answer is: convenience. It's sometimes handy to be able to zip through a quick calculation and get the results in binary, hexadecimal, octal, or whatever. If your computer isn't handy (or someone is playing space invaders on it at the moment), there are questions you can work through if you have a calculator to help.

But make no mistake about it: the sophisticated machines are not indispensable. You can do the job with no calculator at all. You can use a simple four-function unit. You can do useful calculations with a simple programmable unit, entering programs to do the work. Or you can get a "programmer's calculator."

No Calculator

Honest, there are still people out there who add and subtract – and even multiply and divide – without a calculator of any sort. There are programmers who know how to add and subtract in hexadecimal or octal. It's probably good for you to know number systems from firsthand experience.

For example, to convert a decimal number to hexadecimal, divide the number repeatedly by 16. The remainder from each division is a hexadecimal digit; you'll generate the digits from right (low order) to left. So 200 decimal is converted as follows: 200 divided by 16 gives 12 with a remainder of 8. Our last hex digit is 8. Continuing: 12 divided by 16 gives nothing with a remainder of 12. Our next hex digit is 12, which we write as C. The hex value: C8.

Going the other way – from hexadecimal to decimal – is just as easy. We take the digits from the left. After we pick a digit, we see if there are any more. If so, we multiply by 16 and add the value of the next digit. So hex C8 becomes 12x16 +8 or 200 decimal.

On The Computer

It's not hard to write a program to do the conversions. The problem is this: we usually have a program half-written on our machine at the moment we wish to convert something. Loading a program is out; we'd lose our work in progress. For this reason, we usually use direct statements.

From hex to decimal, we usually multiply by powers of 16. Thus, the hex address 027A is evaluated by the direct statement PRINT 0*4096 +2*256 +7*16 +10. 4096 is 16 to the third power; 256 is 16 squared.

From decimal to hex, there's no fixed method. Some people divide the number by 4096 to get the first digit. For example, 59468 divided by 4096 yields 14.5185547 – 14 is a letter E, our first digit. After that, there are a variety of methods: subtracting out the high amount (59468 - 14*4096) is one way, and using the fractional value (.5185547 x 16) is another. In either case, a little work starts to reveal the following digits.

The Four Function Calculator

Most calculators aren't very good at giving you remainders after a division. They will happily tell you that 59486 divided by 16 is 3716.75, rather than that it gives 3716 with a remainder of 12. For this reason, many users like to work decimal to hex conversions from the high-order end.

For a 16-bit number (0 to 65536), divide by 4096. Repeat four times: note the integer value, which is your hex digit; subtract that value to give a fraction; and multiply by 16.

So for 59468 we divide by 4096 to get 14.5185547. Subtract the 14 – that's E, our first digit – and multiply by 16 to get 8.296875. Subtract the 8 – now we have E8 as the start of our hex value – and multiply by 16 to get 4.75. Subtract the 4 – our number is almost complete at E84 – and multiply by 16 one last time. Our final digit will be close to 12, hex C, so we may write our final hexadecimal value as E84C.

Micro Power Bench™

NEW!

- Single Switch Control of CPU and Peripherals
- Built in circuit breaker protects your system
- Four power expansion outlets
- Options: Power Surge Protect and Cooling Fan
- Compatible with IBM, APPLE, TRS-80 and others

SAVE! SAVE! ONLY $79.00

Order Direct
800-343-4311
Master Charge and Visa Accepted

CAB-TEK, Inc.
Riverside St., Nashua NH 03062
CIVILIZING COMPUTERS

MX 80
ACOUSTIC ENCLOSURE $99!

DEALER INQUIRIES INVITED

P.M.P. 2000™

Property Management Program

By T & F SOFTWARE COMPANY

Turn your computer into a property manager with great programs like APARTMENT STATUS REPORT, TENANT STATUS REPORT, CASH RECEIPTS WORKSHEET, INVOICING, DISTRIBUTION OF EXPENSES/BANK ACCOUNT STATEMENT and INCOME/EXPENSE SCHEDULES. P.M.P. 2000 allows you an efficient and low cost way to take care of all your property management needs.

Retail $219.95

Template for the VisiCalc® Program
Available for Atari® 400/800 32K & Apple® II/II plus 64K
Check your local dealer or send check or M.O. with $2.00 postage and handling. California residents add 6½% sales tax.

T & F SOFTWARE COMPANY
A Talcove & Familian Company
10902 Riverside Drive • North Hollywood, CA 91602
(213) 501-5845

SWEET-TALKER, IT GIVES YOUR COMPUTER AN UNLIMITED VOCABULARY.

Parallel Version
Apple II Version

As Featured in Ciarcia's Circuit Cellar, Byte Magazine, September 1981.

The Sweet-Talker voice synthesizer allows you to add speech of unlimited vocabulary to your computer. Utilizing the Votrax SC-01A chip, you can output any message by programming individual phonemes. Comes in two versions; one plugs directly into your Apple II, the other connects to any computer with an 8-bit parallel printer port. + 12 volts and + 5 volts required for parallel board.

- Contains 64 different phonemes accessed by a 6-bit code.
- Automatic and manual inflection modes.
- Parallel port driven or plug-in compatible with Apple II.
- Super text-to-speech algorithm on disk for Apple II. Makes Sweet-Talker equivalent to units 3 times the cost.
- On board audio amplifier
- Sample program on cassette with Apple II board.
- Optional power supply for parallel board.

ST01	Sweet Talker Parallel Port Board A & T........$139.00
ST02	Sweet Talker Apple II plug in board............149.00
ST06	Text-to-Speech algorithm on disk for Apple II...............35.00
SC01A	Votrax Speech Synthesizer chip..........70.00; 5 or more............55.00 each
UPS01	Universal Power Supply-A & T....................35.00

Add $2.00 for shipping & handling.

To Order: Call Toll Free 1-800-645-3479
(In N.Y. State Call: 1-516-374-6793)
For Information Call: 1-516-374-6793

MICROMINT INC.
561 Willow Ave.
Cedarhurst, NY 11516

MICRO MINT
Largest US Distributor of Votrax Chips

NEW GENERAL LEDGER SYSTEM for ATARI* 800

$149.95
Microsoft Base

CHART OF ACCOUNTS
TRIAL BALANCE
INCOME STATEMENT
BALANCE SHEET

$219.95 - includes Microsoft Compiler

VISA - MASTER CHARGE - CHECK - MONEY ORDER

F.C.C. inc.
4712 CHASTANT ST.
METAIRIE, LA. 70002
(504) 454-2421

*Trademark ATARI, INC.

Get More From Your PET/CBM!

NEW! • **DISK-O-MATE™** (Write for Price)
A must for 2040/4040 disk owners. Write protect indicators/switches, power indicator and error beeper.

• **"Real World" SOFTWARE** ($17 - $25)
Word Processor, Mailing List, Catalog, Ham Radio, Frequency Counter.

"OLD" 8K PETs

• **2114-TO-6550 RAM ADAPTER** ($12-$25)
Replace 6550 RAMs with low cost 2114s. *Hundreds Sold!*

• **4K MEMORY EXPANSION** ($16 - $62)
Low cost memory expansion using 2114s for bigger programs.

Write for FREE Catalog!

OPTIMIZED DATA SYSTEMS
Dept. C, P.O. Box 595 - Placentia, CA 92670

DISK-O-MATE trademark Optimized Data Systems -- PET/CBM trademark Commodore

Hexadecimal to decimal is much easier. Take the first digit's value. If there are any more digits, multiply by 16 and add the next digit. Keep going until you have the value. Hex E84C works quickly to its decimal value via these numbers: 14, 232, 3716 and finally 59468.

The Programmed Calculator

With a programmable unit, we can place the above calculations into a program and have the steps done automatically for us.

Many programmable units have a FRAC function which simplifies the sequence of steps. FRAC is the opposite of the INT function. For example, FRAC of 8.296875 yields .296875 and allows us to save a subtraction step in the conversion.

Since most programmable calculators can't input, calculate, and display hexadecimal digits, it is not possible to show (or enter) a value such as E84C. The usual way to overcome this problem is to use a "double digit" hex display, so that E84C will be displayed as 14080412 – the 14 standing for E, the 08 for 8, and so on.

The TI Programmer

The Texas Instruments Programmer is a special-purpose calculator which allows input and display of decimal, hexadecimal, and octal numbers, together with easy conversion between them. Simple four-function arithmetic can be performed, plus logical functions such as AND and OR.

The calculator is not programmable. It has a memory which allows storing a number or accumulating a total. In decimal mode, fractions can be entered – for example, 36.25 – but no fractions can be used in the other number bases.

Relative branch address calculations can be performed by simple subtraction. And the conversions are very simple – just push a button.

The Hewlett-Packard 16C

The H/P 16C is a more expensive calculator, but has many more features. Not only does it have all the logical functions (AND, OR, XOR, and NOT), but it also has an extensive set of Rotate and Shift commands, including a Carry flag. There are commands to set, clear, or test individual bits within a number, and functions which create a "mask" of any number of high bits or low bits.

Conversion of numbers is simple, of course. The 16C will copy with negative numbers, if you wish. You may set it to: unsigned numbers; twos-complement signed numbers (the "usual" way of holding signed numbers); and ones-complement signed numbers, a relatively rare way of representing negative values. We may limit the calculator to a specific number of bits, so that -1 will be shown as hex FFFF in 16 bits or FFFFFF in 24 bits.

The 16C has an "integer" side, with decimal, hexadecimal, octal, and binary display modes; and a "floating" side, which allows decimal numbers complete with fractional parts. The floating mode is good for conventional calculation, although it has no scientific functions.

A remarkable thing about this calculator is that it allows you to convert between floating point numbers and floating binary notation. This is a good trick, since it involves generating an exponent and a mantissa. Not everyone needs this feature, but it's surprising to see such a powerful calculation available.

Floating Point To Decimal

Let's work through this calculation on a variable in Microsoft BASIC. Somewhere in BASIC is a floating-point value stored as hex 81 49 0F DA A2. The 81 is the exponent, and the rest is the mantissa. Let's find its decimal value. Press "f" "2's" to ensure that the machine is in twos-complement signed mode; press "HEX" 30 "f" "WSIZE" to put us into the hexadecimal mode with enough bits to work on.

Now we enter the mantissa: 490FDAA2. Microsoft drops the high bit from positive numbers; we must put it back by pressing "ENTER" 1F "f" "sb" (for set bit 1F, or bit 31). Now we should see C90FDAA2, our corrected mantissa. Now for the exponent: type in the 81. To adjust for differences between Microsoft and Hewlett-Packard, we must subtract hex A0: type "ENTER" A0 "-" (minus).

The display will show something like FFFFFFE1, our adjusted exponent. We're all ready: press "f" -"FLOAT" -8 and we'll see the value: 1.57079633, or one-half pi. We can go the other way just as easily: press "HEX"; adjust the exponent by adding A0; flip to the mantissa by pressing "X-Y"; knock out the high bit by typing 1F "f" "CB". Easy. Remarkable.

The 16C is programmable. The above sequence of operations, or any other, may be entered as programmed instructions so that a simple key sequence (for example, GSB A for GOSUB A) will trigger the whole computation. The calculator has continuous memory; even if it's switched off, the program – and for that matter the data – will remain.

The calculator has many memory locations. How many? That depends on two factors. First, the size of the programs you have stored, if any. Each program instruction takes away from memory space. Second, the "word size" that we have selected. If we decide to work only with eight-bit numbers, for example, we'll have a very large number of memory locations – up to 203. With the maximum size number – 56 bits – or floating point numbers, we get up to 29 memory registers. Up to 32 registers can be accessed from the keyboard, and all registers can be reached via indirect addressing.

Substantial memory plus programming can yield quite powerful systems. It's not too hard to store dozens of 16-bit addresses in a memory table, and look them up as desired. The bit manipulation capabilities can be used to good effect for chip register decoding. Where is the screen and character table for a given VIC configuration? Just type in the appropriate VIC register contents and let the calculator work it out.

Do you need a calculator that's this good? It depends on what kind of work you do. It's an expensive toy, but could be an invaluable work tool.

It's probably good for you to work out things by hand, once or twice. You will understand the mechanism better, and appreciate your calculator/computer more.

Simple calculators or your computer will do number conversion jobs for you nicely at minimum expense. You'll need to remember the proper procedures, but they are not difficult.

The specialized calculators cost more. They do a nice job. You'll have to decide whether the work you do merits the investment. ©

Part I:

Commodore 64 Video — A Guided Tour

Jim Butterfield, Associate Editor

We're about to embark on a guided tour of the 6566 chip, which gives the Commodore 64 its video. It's called the VIC, for Video Interface Chip; that's the same name used for the 6560 chip in the VIC computer, but the 6566 is a whole new story. Along the way we'll stop for lots of experiments, tricks for you to type in to see the effects of manipulating this remarkably versatile part of your computer.

Before setting off on our expedition, we need to establish a few landmarks which will place the chip within the Commodore 64 architecture.

Memory And Video

The 6566 chip relates to memory in two ways. First, the chip's control registers are accessible in addresses 53248 to 53294, or if you'd rather, hexadecimal D000 to D02E. We'll change these registers if we want to change the behavior of the chip.

The chip itself looks directly into memory as it generates video. It is usually looking for at least two things: what characters to display, and how to display them. It finds what characters to display in an area called "screen memory," or, more formally, the "video matrix." It finds out how to display the characters by looking at the "character generator" table, or the "character base."

Since the chip generates a lot of video, it looks at memory a great deal. Most of the time, it can do this without interfering with the processor's use of memory; but every five hundred microseconds or so, it needs to stop the processor briefly in order to get extra information. This doesn't hurt anything: the pause is so short that we don't lose much processing time.

But occasionally, the microprocessor is engaged in timing a critical event and does not want to be interrupted. In this case, it shuts off the 6566 chip until the delicate work is over. Ever wondered why the screen blanks when you read or write cassette tape? To give the computer an extra edge while timing tape, that's why.

Charting The 64

When the video chip goes to memory for its information, it has a special problem: it can reach only 16K of memory. That's OK for most work. For example, the screen (or video matrix) is usually located at 1024 to 2023 (hex 0400 to 07E7), so we'll use it there. But if we wanted to move screen memory to a new location, say 33792, we would need to work out some details, since the chip would not normally be able to reach addresses so high in memory.

We are given some help in doing this by the 64 architecture itself. There are two control lines called VA15 and VA14 which allow us to select which block of 16K memory we want the video chip to use. Note that once we've selected a block, the chip must get all its information from that block: we can't mix and match.

The control lines are available in address 56576 (hex DD00) as the two low-order bits. The memory maps you get are:

- **POKE 56576,4** the chip sees RAM from 49152 to 65535. There's no character generator; you'll have to make your own.
- **POKE 56576,5** the chip sees RAM from 32768 to 36863 and from 40960 to 49151. The ROM character generator is in the slot from 36864 to 40959.
- **POKE 56576,6** the chip sees RAM from 16384 to 32767. No character generator.
- **POKE 56576,7** the chip sees RAM from 0 to 4095, and from 8192 to 16383. The ROM character generator is in the slot from 4096 to 8191. This is the normal Commodore 64 setup.

Also note that the chip never has access to RAM at addresses 4096 to 8191 and 36864 to 40959. You will not be able to put screen memory or sprites there.

Be careful with these. If you move the chip's memory area, you'd better be sure to move the screen. For example, try the following:

```
POKE 648,132:POKE 56576,5
```

You'll find yourself transferred to a new, alternate screen. The new screen will be "dirty" – it hasn't been cleaned up. Typing a screen clear will make things look neat, and you may then play around with an apparently normal machine. When you're finished, turn the power off for a moment to restore your machine to the standard configuration.

The Chip: Video Control

Now for the 6566 chip itself. We'll go through the registers, but not in strict numeric order.

Table 1:

6566 Video Chip
C64 Control and Miscellaneous Registers

Register	Bit fields								Decimal
D011		Extended Color Mode	Bit Map	Display Enable	Row Select	Y-Scroll			53265
D012	Raster Register								53266
D013	Light Pen Input X								53267
D014	Light Pen Input Y								53268
D016	X	X	Reset	Multi Color	Col Select	X-Scroll			53270
D018	Screen: VM13	VM12	VM11	VM10	Character Base: CB13	CB12	CB11	X	53272
D019	IRQ	Interrupt ←Sense→			LP	SSC	SBC	RST	53273
D01A	Interrupt Enable →				Light Pen	Sprite Collision with Sprite	Sprite Collision with Back	Raster	53274

Color Registers

Register			Decimal
D020	X	Exterior	53280
D021	X	Background #0	53281
D022	X	Background #1	53282
D023	X	Background #2	53283
D024	X	Background #3	53284
D025	X	Sprite Multicolor #0	53285
D026	X	Sprite Multicolor #1	53286

Table 2:

6566 Video Chip
C64 Sprite Registers

Sprite 0	Sprite 7			Sprite 0	Sprite 7
D000	D00E	Position X		53248	53262
D001	D00F	Position Y		53249	53263
D027	D02E	X	Color	53287	53294

Sprite: 7 6 5 4 3 2 1 0

Register	Function	Decimal
D010	X-Position High	53264
D015	Sprite Enable	53269
D017	Y-Expand	53271
D01B	Background Priority	53275
D01C	Multicolor	53276
D01D	X-Expand	53277
D01E	Interrupt: Sprite Collision	53278
D01F	Interrupt: Background Collision	53279

Location 53265 (hex D011) is an important control location. It contains many functions; its normal value is 27 decimal.

Values from 24 to 31 control the vertical positioning of the characters on the screen. Try this:

```
FOR J=24 TO 31:POKE 53265,J:NEXT J
```

You'll see the screen move vertically, leaving an empty spot near the top. POKE 53265 back to 27.

If we subtract 8 from the value in the 6566, the screen will lose a line: instead of 25 lines we'll have only 24. The best way to see this is: clear the screen; write TOP on the top line, BOTTOM on the bottom line (don't press RETURN!) and then move the cursor to about the middle of the screen and type:

```
POKE 53265,19
```

You'll see the top and bottom trimmed to half a line each.

Think about using these two features together. If we have a screen full of information, we would normally scroll when we wanted to write more – the characters would jump up a line. But if we can switch to 24 lines, slide the characters up gently, and then switch back to 25 lines, we'd have a smoo-ooth scroll.

```
POKE 53265 back to 27
```

If we subtract 16 from this location, we'll blank the screen. We mentioned this before: it will give the processor a little more accuracy in timing. In fact, this POKE is the key to allowing us to LOAD a program from an old-style 1540 disk unit. If the disk hasn't been modified, it will deliver bits slightly too fast for the computer. But we can bridge the gap with POKE 53265,11:LOAD and the loading will take place successfully. When the load is complete, we can get the screen back with POKE 53265,27.

High Resolution

The next control bit – value 32 – switches the display to pure bits. No more characters: the screen will be purely pixels as we switch to high resolution mode. We'll use a lot of memory for this one: memory to feed the screen will be 8000 bytes.

High resolution needs to be carefully set up, but let's plunge right into it. Type POKE 53265,59 and you'll see an intricate pattern on the screen. What you are looking at now is a bit map of RAM memory addresses 0 to 4096, plus the character generator area. The top of the screen will twinkle a little: those are the page zero values changing – things like the realtime clock and the interrupt values are constantly in motion.

In the bottom half of the screen, we'll see the character generator itself. Oddly enough, the characters are readable. That's because of the way high resolution bit mapping works: each sequence of eight consecutive bytes maps into a character space, not across the screen, as you might think.

Now we're going to play around a little. First, clear the screen. Surprise! It doesn't clear, but the colors change. That's because screen memory, into which we are typing, holds color information for the high resolution screen. Now, we'll clean out a band of hi-res data by typing in a BASIC line. We must do this "blind"; the screen won't help us. Type:

```
FOR J=3200 TO 3519:POKE J,0:NEXT J
```

If you've typed correctly, you'll see a blank band across the screen. Don't worry about the color change as you type. Now we'll enter (blind again):

```
FOR J=3204 TO 3519 STEP 8:POKE J,255:NEXT J
```

You should see a high-resolution line drawn across the screen.

That's all the high resolution fun we're going to have this session, but you may be starting to get an idea of what's going on. Turn off the power, and let's look at other things.

Extended Color

If we add 64 to the contents of 53265, we'll invoke the extended color mode. This will allow us to choose both background and foreground colors for each character. Normally, we may only choose the foreground: the background stays the same throughout the screen. You lose some colors, but get better combinations.

Try POKE 35265,91. Nothing happens, except that the cursor disappears, or at least becomes less visible. Why? We've traded the screen reverse feature for a new background color. Try typing characters in reverse font, and see what happens. Try choosing some of the specialized colors – the ones you generate with the "Commodore" key rather than CTRL. See how you like the effect. Think how you might be able to use it.

Extended color is purely a screen display phenomenon. POKE 35265,27 will bring all the characters you have typed back to their normal appearance.

The High Bit

There's one more bit in location 53265, the one we would get if we added 128. Don't do this now: this bit is part of a value we'll discuss later: the "raster value." You won't use this one out of BASIC, but it can be handy at machine language speeds.

Tune In Again

We've done a lot of things so far, using only one control location. There are more locations, and we'll discuss some of them next time.

It's a big chip. It will take a lot of time to digest all its possibilities. It's fun, and it can create remarkable effects. ©

Bi-directional VIC Scrolling

Charles Saraceno

How would you like to be able to check and debug your VIC programs by turning your screen into a window which can move anywhere over the listing, stop or start at will, and even move upwards toward the start of the program? All this can be achieved by just touching different keys when using this clever "controlled scrolling" program. If your VIC has the 3K RAM memory expander plugged in, use POKE 44,4 (instead of POKE 44,16) in the instructions in the final paragraph.

Now that memory expansion modules are readily available, it is possible to write longer VIC programs. This does make it harder, however, to edit the contents without a hard copy from a printer to examine for typing errors. Screen editing is time consuming, to say the least; with 22 characters per line, you are limited to four or five lines at a time between LIST commands. A very useful LIST would scroll the screen and stop or continue when you want it to. The ideal LIST would also scroll backwards.

This small program efficiently accomplishes all these tasks. Line 63001 determines the starting address (SA) for any memory installed into the VIC. Line 63002 calculates the line number (LN) of your program. Line 63003 sets your screen up to perform the tasks needed to list the line, then continues the program. It is written in white so you won't see the commands and keeps the screen uncluttered for reviewing the listed line.

Once a "list" has been initiated in a program, the program will end. This is where the keyboard buffer commands in line 63004 both control the list and then continue the program with the "go to" 63010 command. Lines 63010-63030 let you review the line just listed and wait for you to press the "+" key to advance to the next line or the "-" key to back up to the previous lines listed. Line 63100 looks for the next "0" in BASIC, which indicates the end of that BASIC line, and then sends you back to calculate the next line number. Line 63200 is the routine that looks for the end of the previous line. You have to eliminate the possibility of finding a "0" in the addresses that determine the line number by disallowing a "0" in either of those two addresses.

One other little trick will let you avoid having to type in this program after each main program has been entered. Find the end of BASIC by typing in:

CLR: PRINT PEEK (45), :PRINT PEEK (46)

Now type the following line which moves the beginning of BASIC to two bytes less than the end of the program (either a null or a "0" is needed to start loading in a new program):

POKE 43, PEEK (45)-2:POKE 44,PEEK (46)

Now load in "+/- LIST" program, reset BASIC pointers (POKE 43,1; POKE 44,16, for VIC with no expansion). Start editing by typing in RUN 63000. You will be able to scrutinize your program on a line-by-line basis. Any mistakes discovered should be noted on paper and corrected after your review.

```
63ØØØ REM**  +/- LIST   **
63ØØ1 SA=PEEK(44)*256+PEEK(43)-1
63ØØ2 LN=PEEK(SA+3)+PEEK(SA+4)*256
63ØØ3 PRINT"{CLEAR}{WHT}GOTO 63Ø1Ø":PRINT"LIST
      ";LN;
63ØØ4 POKE631,19:POKE632,17:POKE633,31:POKE634
      ,13:POKE 635,19:POKE636,13:POKE 198,6:E
      ND
63Ø1Ø IF PEEK(197)=5 THEN 631ØØ:REM TEST FOR "
      -" KEY
63Ø2Ø IF PEEK(197)=61 THEN 632ØØ:REM TEST FOR
      "+" KEY
63Ø3Ø GOTO 63Ø1Ø
631ØØ IF PEEK(SA+5)<>Ø THEN SA=SA+1:GOTO 631ØØ
6311Ø SA=SA+5:GOTO 63ØØ2
632ØØ SA=SA-1:IF PEEK(SA)=Ø AND PEEK(SA-4)<>Ø
      AND PEEK(SA-3)<>Ø THEN 63ØØ2
6321Ø GOTO 632ØØ
```

INSIGHT: Atari

Bill Wilkinson

This month we will examine the possibility of a "default" drive number under DOS. There is also a tidbit about initializing DOS disks from BASIC. Next month, we will begin what will be a three- or four-month series on how to write your own BASIC interpreter.

Deriving The Drive

First, let me state that I do *not* recommend this section to the relative novice. While it is true that you can perform the operations I am about to describe entirely from BASIC, it is also true that you can destroy memory very nicely if you slip up. Enough warning. To begin:

Have you ever (often?) grumbled over the fact that you have to specify not only the file name, but also the disk name and drive number (e.g., "D2:MISSILE.CMD")? I sure have. In fact, I hate it so much that when we did OS/A+ for the Apple II, we allowed the user to supply a default device specifier (e.g., "D2:"), which is automatically prefixed to all file names which do not specify a device. (Consequence: you *must* use a colon when you really want a device; "P" is seen as "D2:P", though "P:" works fine.)

This concept is not new or unique; even in the micro world, such giants as CP/M use default drive assignments. Usually, the advantage of such defaults is that people with multiple disk systems need not always run a given program in a certain drive. Or the user might choose which drive will receive his data files via a simple set of keystrokes at system powerup. Suffice it to say that those who get used to default drives love them.

Unfortunately, as much as I would like to do the same thing for the Atari, I can't. The initial device name determination under Atari's OS is done in the OS ROMs, and Atari OS simply looks at the first letter of any file name and assumes that it is the device name.

However... (You knew there was a "however" lurking, didn't you?) At least we could modify the File Manager System (also known as FMS, DOS, or even OS/A+) to understand the concept of a default device NUMBER. In other words, we could have the FMS inspect the file name and assume a particular drive number if "D:..." were coded. Then we could have some means of telling the FMS what the "current" drive was (and, in fact, such means already exist in OS/A+), and the system would automatically insert the correct drive number.

And yet, I am reluctant to adapt such an approach with Atari DOS. Too many programs have been written which assume that "D:..." is equivalent to "D1:...", and I am loath to introduce more confusion than is necessary. So, if you really would like to modify your copy (copies?) of FMS to allow "D:" to represent "Dn:", let me just point you in the right direction. For this purpose, I will presume that you have a copy of *Inside Atari DOS* (**COMPUTE! Books**, 1982).

There is a routine labeled FNDCODE (File Name DeCODE) which begins on page 83 of the book and is the heart of the entire disk file name processing. Lines 4101 through 4106 start at the third character of the name and search from there backwards for the colon (':') which terminates the device specifier (and ignore the comments in the listing...they are flat out irrelevant). Obviously, it would be no big deal to check to see if the character before the colon is the 'D' and, if so, assign a default device number.

Changing FMS

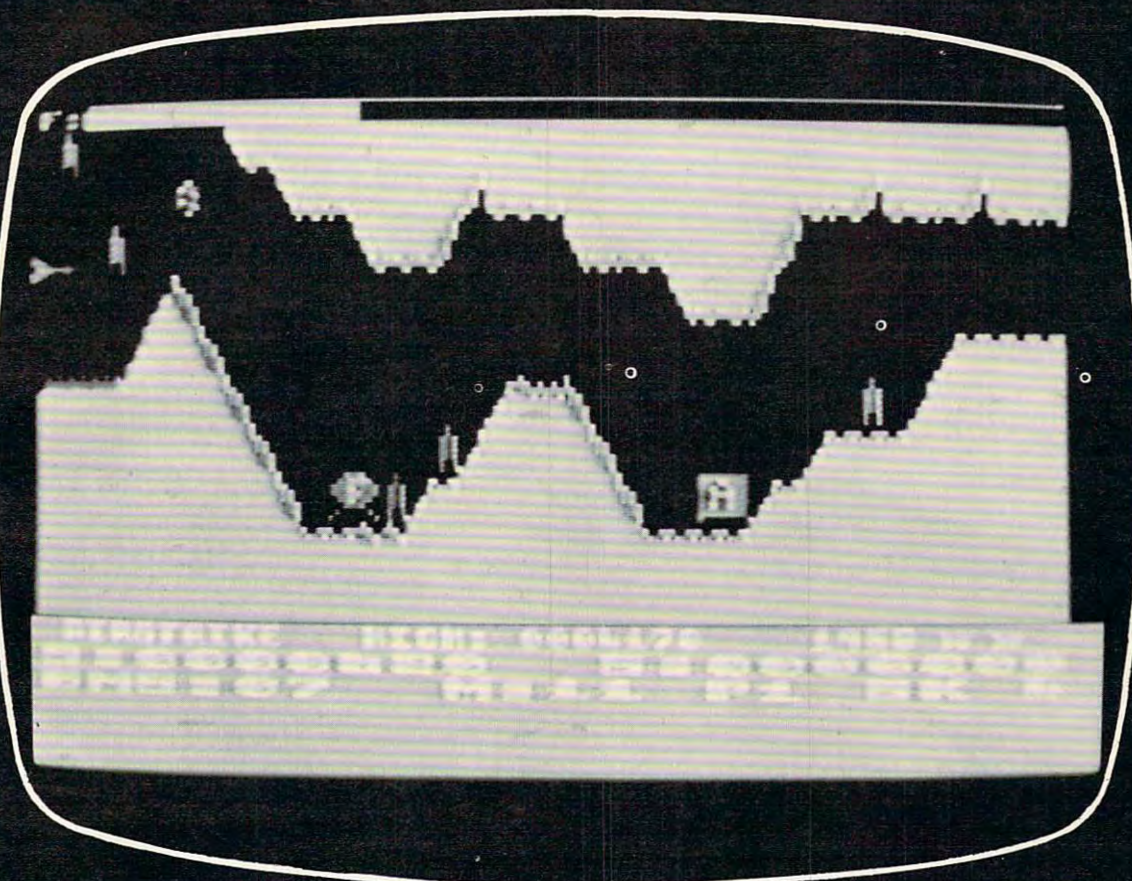
Now, for the rest of you, I have an alternate proposal. How about changing FMS so that, if it sees a file name of "D0:..." it assigns the default device instead. I chose "D0:" because there should be no conflict with existing software. And, yet, it is a legal device specifier which is easily detectable and changeable.

Since the OS ROMs have already decoded the device number by the time FMS gets control, we don't need to look at the file *name* at all. Instead, we look at the field labeled ICDNO (or, in zero page, ICDNOZ), the device number as set up by the OS ROMs. And, conveniently, FMS is already manipulating this number in a single, well-defined place, the "SETUP" routine (as listed on page 92 of *Inside Atari DOS*). Currently, the code sequence is simply:

```
LDY ICDNOZ ; move device number...
STY DCBDRV ; ...to device control block
```

What we want instead is something like the following:

Very, very, difficult!



The definitive, super-fast, multiple skill, shoot-out game for Atari® 400/800™ with superb arcade-type realism.

Retail price for 16K cassette or disk: $39.95
One or two player game.
Coming soon: Venus Voyager, TimeWarp, and more programs by fine English software designers and authors.

The English Software Company
P.O. BOX 3185 • REDONDO BEACH, CALIFORNIA 90277
(213) 372-3440

ENGLISH SOFTWARE™

```
   LDY    ICDNOZ      ;get device number...
   BNE    OKDNO       ;...if wasn't "D0:", it is OK
   LDY    DEFAULT     ;otherwise, change 0 to default
OKDNO
   STY    DCBDRV      ;in either case, set up DCB
```

Now I can't think of a much simpler change than adding two instructions, but how do we make such a change? The solution is to use what is known as a "patch." Generally, there are two kinds of machine language patches: those that fit into the original code space and those that don't. The former kind are easy; simply overlay the old code with the new. The latter are not so easy. Naturally, this change falls into the latter category.

With a 6502, the usual method of installing out-of-line patches is to try to replace a three-byte instruction with a JMP or JSR to the patch (failing this, you must replace two or three instructions, which may involve putting a NOP before or after the JSR or JMP). Luckily, we do indeed have a three-byte instruction that we can replace (the STY DCBDRV uses three bytes, since DCBDRV is not in zero page).

So our patch will look like this:

```
DCBDRV  = $301        ;object of the STY
   *=     $1176       ;address of the STY instruction
   JSR    PATCH

   *=     PATCH
   BNE    OKDNO       ;non-zero device number
   LDY    DEFAULT     ;replace zero device number
OKDNO
   STY    DCBDRV      ;the patched-over instruction
   RTS
```

So far, so good. It makes sense, I hope. But there are two locations undefined in the above listing: we don't know where PATCH and DEFAULT are going to be located. Again, we will refer to the book for some clues as to where they should be.

As it turns out, there is no patch space at all within the main code space of FMS. However, if we look at the very end of the listing (page 98 in the book), we find that FMS (including its internal buffers, etc.) ends at $1500. But remember that "DOS.SYS" consists of more than just FMS. In the case of OS/A+, DOS also includes "CP," the console processor, and actually ends at $1D00. For Atari DOS, version 2.0S, DOS.SYS ends at $1A7C (to accommodate "MINI-DUP," the routine which handles MEM.SAV and loads the main DUP.SYS).

But, fortuitiously, whether by design or by chance, both MINI-DUP and CP begin at $1540. Thus, we have locations $1501 through $153F for patch space. Not a huge patch space, but patch space nevertheless. So, I would suggest that you add the following two lines to the front of the listing given above:

```
   DEFAULT = $1501
   PATCH = $1502
```

This means, then, that you *must* put a valid disk drive number (1 through the number of drives you have) into location $1501 *before* using a drive specifier of "D0:".

So, how do we make and save this patch? If you have an assembler capable of doing memory-to-memory assemblies (e.g., the cartridge, EASMD, MAC/65, etc.), I would suggest typing in the lines given and actually assembling the code directly in place. (Doing the memory-to-memory assembly avoids doing FMS accesses while patching FMS...safety first!) Then, with the patch in place, use the Write-DOS-Files option (of Atari DOS, or use INIT to rewrite DOS.SYS with OS/A+) to save your patched system.

Does it work? Sure does. I wrote all the above and then went over to the machine and typed it in. Worked first time! Is it handy? Only time will tell.

And one more point. If you do have OS/A+, you will note that the Command Processor (CP) already supports the concept of a default drive. Why not use that same default drive specifier for our "D0:" trick? The only difference is that CP stores that default specifier as an ASCII character ('1', '2', etc.), so we must look at only the low order bits of the default (and we must obtain it from its memory location according to OS/A+ rules). So here's another version of the same patch, specifically for OS/A+, version 2:

```
PATCH    = $1501
CPALOC   = $0A
DEFAULT  = 8
DCBDRV   = $301
  *=     $1176
  JSR    PATCH
  *=     PATCH
  BNE    OKDNO         ;drive # is non-zero
  LDY    #DEFAULT      ;offset to default drive #
  LDA    (CPALOC),Y    ;gets default in ASCII
  AND    #$0F          ;just the lower bits
  TAY                  ;where FMS expects drive #
OKDNO
  STY    DCBDRV        ;the patched-over code
  RTS                  ;back to the original
```

And, as a postscript to all this, I would like to comment on the whole subject of adding things to DOS. So long as you can patch in place or use the limited patch space starting at $1501, you should have no problems. If, however, you want to add significant code to DOS, it will not be easy if you are using Atari DOS.

If we look at pages 94 and 95 of *Inside Atari DOS*, we will see the routine which begins with the label "WD0". It is this routine which actually writes the file "DOS.SYS" to the disk. And, if you look at lines 5441 through 5449, you will see that what is written out is all of memory from $7CB

ATARI
A Warner Communications Company

MEMORY

FOR THE ATARI® 400/800

ATARI
A Warner Communications Company

Alpha Computer Center
West Seneca, New York
716-674-5511

Computer Concepts
Cedarhurst, New York
516-374-0255

Comtech
Brooklyn, New York
212-332-5933

S.T.C. Marketing
Richmond Hill, New York
212-848-8878

Tri-Comp H.C.S.
Bayport, New York
516-472-9769

Video Village Inc.
Massapequa Park, New York
516-779-1616

Computer Land
Charlotte, North Carolina
704-536-8500

A B Computers
Colmar, Pennsylvania
215-822-7727

Computer Terminal
Summerville, South Carolina
803-875-4727

Hi-Fi Buys
Springfield, Virginia
703-644-5252

H.E.L.
Fairfax, Virginia
703-698-8595

ASMUS Electronics Inc.
Fond Du Lac, Wisconsin
414-923-4107

Madison Books & Computers
Madison, Alabama
205-772-9250

All Systems Go
Tempe, Arizona
602-966-4275

48K *$124.95 (With 16K in trade)

32K $94.95

2 YEAR FULL WARRANTY

THESE BOARDS DO EVERYTHING THE COMPETITORS WILL DO — BUT COST YOU MUCH LESS.

48K (Without 16K trade) **$149.95**

32K LOOPBACK — Allows 32K to be used alone in the 800 .... **$5.00**

48K LOOPBACK — Allows 48K to be used alone in the 800 .... **$5.00**

Call JERSEY SYSTEMS

OR YOUR LOCAL PARTICIPATING DEALER

Data Concepts
Phoenix, Arizona
602-242-9961

Data Concepts
Scottsdale, Arizona
602-990-3366

Leisure Electronics
Littlerock, Arkansas
501-227-4837

H.C.M.S.
Westminster, Colorado
303-427-9036

Sound Trader and
Computer Corner
Tampa, Florida
813-977-4868

Computech
Tallahassee, Florida 32312
904-893-1743

Computer One
Indianapolis, Indiana
317-259-4827

The Computer Shop
South Bend, Indiana
219-232-9918

New Generation
Baton Rouge, Louisiana
504-272-1700

World Wide Home Video
Metairie, Louisiana
504-456-9926

Software Asylum Inc.
Carteret, New Jersey
201-969-1900

Video Home Center
Manalapan, New Jersey
201-431-7636

A & B Software
Union, New Jersey
201-575-5331

- TO ORDER CALL: **800-526-3647**
- NO CHARGE FOR SHIPPING
- ORDERS SHIPPED U.P.S. BLUE LABEL WHEN POSSIBLE
- 2% CREDIT FOR ORDERS NOT SHIPPED IN 24 HRS.

FROM NEWELL INDUSTRIES:

FASTCHIP
- Up to 3 and one half times the speed of the original floating point routines.
- Pin compatible masked ROM — Permanent replacement for the original ROM.
- No modifications, cuts or wires.
- 90 DAY FULL WARRANTY
- Fits 400/800

— $41.95 —

JERSEY SYSTEMS
P.O. BOX 332
EDISON, N.J. 08818

☐ CHARGE MY (master charge) (VISA)
EXP. DATE ____________
CARD NO. ____________

☐ ______ 48K BOARD $149.95 ☐ ________ 32K BOARD $94.95

☐ ______ FASTCHIP $41.95 ☐ MORE INFORMATION

N.J. RESIDENTS ADD 5% SALES TAX, PLEASE

NAME ____________

ADDRESS ____________

PHONE NUMBER ____________

* PARTICIPATING DEALERS ONLY

® Trademark of Atari Inc.

through the contents of location "SASA" (which are usually $1A7C or $1D00, as noted above).

Sidelight: in a way, this is poor design, since SASA also specifies the beginning address of the disk buffers. If you move the disk buffers (e.g., to the top of memory) and then try to write the DOS file(s), you might be writing out much more than you bargained for. You might want to change those compares to

"CMP #..."

if you are doing hefty modifications.

Anyway, with Atari DOS, you can't really add on to the end of the DOS.SYS since DUP.SYS begins immediately after it in memory and would overwrite your additions. With OS/A+, though, you could add stuff at $1D00 (or wherever SASA points to) and move SASA up (which not incidentally will thus move the buffers out of the way of your addition).

The Rites For Right Writes

I was reminded by all of the above of another "feature" of Atari DOS (and, yes, OS/A+) which is not well documented. In particular, would you like your program (including one written in BASIC) to be able to write (or rewrite) the "DOS.SYS" file? In the unlikely case that your answer is "yes," read on.

Strange but true: when you OPEN the file named "DOS.SYS" for output (i.e., mode 8 only), *right then and there* the FMS will automatically write the complete boot (sectors 1, 2, and 3) and the file "DOS.SYS" to the disk! You do *not* have to copy anything from memory to disk, from disk to disk, or what have you. FMS does it all! (And that explains why Atari DOS won't let you copy *to* a file called "DOS.SYS".)

Thus, from BASIC, you could initialize a disk AND write the DOS.SYS file via the following simple code:

```
10 XIO 254,#1,0,0, "Dn:"
20 OPEN #1,8,0,"Dn:DOS.SYS"
30 CLOSE #1
```

Of course, the "n" can be any valid disk number (including 0, if you applied the patches discussed in the first section of this column). Also, you can omit line 10 if you don't want to initialize the disk.

Unfortunately, this procedure will not place "DUP.SYS" on the disk if you are using Atari DOS, so you will still have to somehow copy it. (But you can use AUTORUN.SYS based systems without DUP.SYS, of course.) Again, though, if you are using OS/A+ you don't (and can't) use a DUP.SYS file, so the above little program will perform all you need to initialize a master, bootable disk.

Postscript: If you really *need* to copy a "DOS.SYS" file from one disk to another (because, for example, you don't want to boot the version that you are copying), you can simply rename "DOS.SYS" to something else ("GORP.SYS", for example), perform the copy, and then rename both the old and new "GORP.SYS" back to "DOS.SYS". Thanks to the peculiarities of FMS, this method will even cause the three boot sectors to be updated to point to your new DOS file. ©

Jim Butterfield, Associate Editor

The New 6500 Chips

The 6502 is a member of a family of chips. The original family included the 6501 (long since extinct), the 6502, 6503, 6504, 6505, 6506, and 6507. A parallel branch of the family comprised the 6512, 6513, 6514, and 6515; these were identical to their 650x counterparts except for the external clock circuitry.

The 6502 is the big member of the family; it has a full 40 pins. The 6503 to 6507 are cut-down versions of the same chip, with only 28 pins. Internally, the chips are the same: the programmer will use exactly the same instructions regardless of which chip is involved. The practical difference is how the chip is wired, and how much memory it is able to address.

If the same chip goes into a 6502 and, say, a 6504, why not take the fully-featured processor every time? The answer is this: if you don't need the extra pins, you can save money by going for the small one. Process controllers often need very little memory; savings in board space and a lesser number of connections can be quite worthwhile.

Quick And Easy

The 6502 burst onto the microprocessor scene in 1976. It was remarkably inexpensive and seemed to have a very simple internal structure. The architecture was closest to Motorola's 6800 microprocessor series, and many users suspected that the 6502 was a cheap imitation. This proved to be untrue: the 6502 had special features which made it a landmark in microprocessor design.

The technique which gave the 6502 speed is called "pipelining." It means that information rolls into the processor as if it were on a conveyer belt. Before the last piece of information is digested, the next one is coming in. For the first time, the microprocessor didn't need to "stop and think": new information was rolling in as the old was being digested. The result: no wasted memory cycles, and amazing speed.

The small number of registers within the 6502 seemed to be a limitation. It proved not to be: registers could be loaded and used so quickly that the small number seldom gave problems. In addition, page zero of memory could be used to hold 16-bit pointers for "indirect addressing" – in a sense, this provided an extra 128 registers for the programmer's use.

The 6502 used the same style of instructions as the 6800 – the simple, traditional data processing instructions: load, store, add, and test. Programmers found the instructions easy and natural. The 6502 is relatively easy to program.

The New Processors

Recently, new 6500-family processors have come into production. They are still familiar: the instruction set is the same as before and the addressing modes haven't changed. But there are new features, and you'll be meeting them in the VIC and in forthcoming Commodore products.

The 6510

The 6510 is a 6502, except that addresses 0000 and 0001 have special functions. There's an input/output port built into the chip: eight pins marked P0 to P7 are available on the microprocessor chip itself. Address 0000 is used as the direction register of the I/O port, and 0001 is the port itself. Otherwise, the 6510 is identical to a 6502.

What does this mean in the Commodore 64? First of all, locations 0000 and 0001 are no longer RAM. PET uses these locations to hold the USR jump; on the Commodore 64, this jump has been moved to address hex 0310 (784 decimal).

Second, you may use address 0001 to test and control some of the 64's activities. Refer to the memory map in **COMPUTE!**, October 1982, for details. For example, you can sense if the cassette tape switch is down by checking PEEK(1) AND 16. The three lowest-order bits are used for switching out ROM and switching in RAM. Don't ever do this from BASIC, and use prudence if you do it from machine language. More on these bits in a moment.

A little more information on memory control from address 0001: bit 0, mask 1, controls the BASIC ROM in addresses A000-BFFF. Switch this bit to zero and the BASIC ROM is gone: in its place is RAM. Now you can write your own language. Bit 1, mask 2, controls the Kernal ROM in addresses E000-FFFF. Switch this bit to zero and the Kernal is gone; be very careful, since you've just switched away all of the programs that support interrupts, keyboard, screen, and so on. If you switch off both bits 0 and 1, you will get a 64K RAM machine: the I/O block will be switched out, too.

The 6509

The 6509, too, is a 6502 with a change to addresses 0000 and 0001. In this case, the changes are more profound: they cause a switch to a new memory bank. The 6509 is expected to be used in the newest CBM products: the PET II (P128) and the CBM II (B and BX series).

Both addresses 0000 and 0001 are used to provide access to memory beyond the normal 64K limitation. These addresses are used to "bank switch" to one of 16 memory banks, each of which is 64K in size. Thus, the 6509 can access over one million memory locations.

If we place a value of zero to 15 in address 0001, we will influence only one kind of address: indirect, indexed. So if we code LDA #$01:STA $01 we are selecting bank one for indirect addressing. Now, if we code LDA ($F0),Y we will perform the following steps: go to addresses 00F0 and 00F1 in the current bank and get the new address stored there; add the contents of the Y register to this new address; and finally, load the A register with the contents of the resulting address, from bank one. Indirect addressing is generally used to obtain or store data; the extra capability provided with address 0001 allows us to obtain or store a very large amount of data.

Address 0000 changes the bank from which we obtain instructions. If we code LDA #$01:STA $00 we will immediately start executing instructions from bank one. This is tricky: we have not jumped, so we will start executing from precisely the same address we left in the other bank. We must carefully write "synchronized" programs so that when we leave one bank, there will be a program in exactly the right place in the new bank to allow processing to continue. It's a good trick, but it can be done.

The new chips are still 6500 style. They use the same instructions in exactly the same way. But they open up new possibilities, and we'll need to learn how to cope with them. ©

PET Dynamic Bookkeeping

Ron Kushnier

This bookkeeping program saves data with a program on tape. Using the "dynamic keyboard" technique and other modifications, it illustrates a useful subroutine for those who want extra power from a tape-based system.

It was my objective, after several years of enjoying my 8K PET, to create a program which I could incorporate into my everyday activities at home. I decided on the proverbial bookkeeping program. Entering all those numbers was a job of which I was not overly fond, but it was all too necessary for income taxes.

This program would have to be practical. It would have to be fast, with easy access, and it would have to do more than could be done by just entering the same information on 3x5 cards.

I tried to consider all the options. Sure, it would be easy if I bought a disk system and something like VisiCalc, but that wouldn't be much of a challenge.

Trial And Error

I took stock of what I had available: standard 8K PET with Upgrade ROMs, and a ROM Toolkit mounted on a PC board which plugged into the PET Expansion socket.

The first thing I did was to buy the Rabbit ROM from Eastern House Software. This high speed cassette firmware improved my access time tremendously. By making a simple modification to the Toolkit board, I was able to change the address of the existing extra socket to that of the Rabbit ROM(A000).

But I ran into two problems. The first was that the Rabbit would not work with the PET's internal tape drive. And since Commodore, at that time, could not supply me with an updated PC board replacement, it was necessary to switch to a whole new CBM cassette unit. The second problem was that the Rabbit could not be used for data files. This meant that I would have to do something really tricky.

The original program was stored with the Rabbit. The data was stored with conventional Commodore data files on another tape. No good! *It took forever*! The time had come for the tricky part.

Again, I explored the options. I could convert array information to data statements. It had been done before. But there was the conversion time both back and forth. The idea was unappealing.

I started fooling around with a dynamic keyboard approach, but without much success. The dynamic keyboard is a method that lets the computer modify the program in memory by POKEing to the keyboard buffer. See lines 58000 on.

It was not until a co-worker, Howard Bicking, came along that the solution was found. He managed to write a small tag-along routine that could be added, which would save all variables and array data along with the program. This was a real breakthrough. I was now able to save 10K worth of program and data in under one minute.

The bookkeeping part of the program is fairly straightforward. It consists of menu-driven, nested arrays of information which allow for easy update and display. Some protection routines were written into the program so that mistakes could be easily corrected.

Obviously, the program must be tailored to the individual user. Its modular construction should make this a fairly easy job.

The Special SAVE

There is a little procedure which must be followed when using the SAVE routine.

After you have entered the bookkeeping program or a modification of the program, it is necessary to run it in order to initially set up the various parameters and pointers. Start the run after the 0 statement number. This is accomplished with a:

```
RUN10
```

Then just follow the program instructions printed on the screen. When all the data is entered, you will want to do a SAVE. The dynamic keyboard will take over, change pointers, and will then display that it is OK to save in a conventional manner.

The next time the program is to be used, a normal RUN will bring in the works.

The SAVE program can be added to any program as the last thing to be done. As a result, self-learning programs can be saved with an ever-increasing library of entries.

```
0 GOSUB59010:GOTO1000
10 DIM Q(3,6,12),M$(12)
1000 PRINT"{CLEAR}";"BOOKKEEPING PROGRAM
1010 FORH=1TO500
1020 NEXTH
1030 GA$(0)="CASH
1040 GA$(1)="AMOCO
1050 GA$(2)="ARCO
1060 GA$(3)="GULF
1070 GA$(4)="SHELL
```