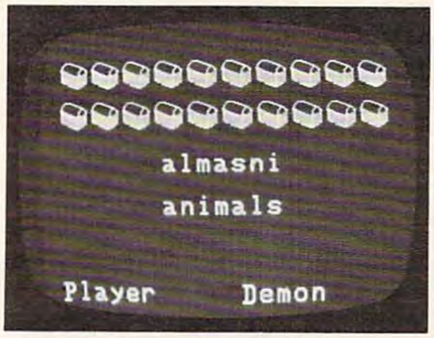
ally hear the words. A second way would be for each word to flash on the screen for a long enough time to be read, and then be erased before the child begins typing it in. A third possibility would be for the program to generate sentences with the target word omitted.

Some of the word lists on the diskette contain words that children misspell because they choose the wrong homonym. Words like "there" and "their" are in the lists, as are "to", "two", and "too". Avoiding these misspellings would be helped only with practice using them in sentences. Unscrambling their letters is of no value whatsoever.

A relatively minor problem with the program is the length of the game. Each game could take up to 40 minutes. This is longer than the amount of time children typically have on a computer at school. If a child needs to stop playing before the game is over, the ESC key will allow an escape from the game. However, when this option is exercised, no score summary is presented, and no reward is given.



```
almasni
animals

Player      Demon
```

Unscrambling spelling words in Magic Spells.

The documentation for this diskette gives information about the game in an interesting, clear fashion. Screen photographs add a lot to its understandability. It gets a little confusing when it gives instructions for making backup copies of lists of words, however. The major confusion is whether the manual is referring to the *Magic Spells* backup diskette or another backup diskette. A teacher or parent new to the computer would probably need some help understanding to which disk the instructions are referring.

This program has potential, but I feel it should be revised. Although the kids love the game, the teacher utility works well, and the flow of the program is appealing and makes logical sense, it should be reworked to include an educationally sound way of giving practice in spelling words.

The program, written by Leslie Grimm, comes with a master and a backup and runs on an Apple II Plus with 48K.

Magic Spells
Apple Computer, Inc.
20525 Mariani Avenue
Cupertino, CA 95014
$45 ©

Frogger For Atari

Larry Isaacs

A number of arcade games have been converted for home computers. *Frogger* has now joined these ranks. We Atari 400/800 owners are fortunate to receive another excellent conversion. This game is sold by Sierra On-Line, Inc., under license from Sega, the same folks who made the original arcade version. The Atari version is credited to John Harris. *Frogger* is available on disk (requires 32K) or cassette.

For those who do not frequent the arcade game rooms, a detailed description of the game follows. But first, there are a couple of options that may be set. Once the game has finished loading, you may choose between two speeds, FAST and SLOW, and whether you want the accompanying music on or off. The OPTION switch toggles the game speed between FAST and SLOW, with FAST being the initial setting. So far, the SLOW speed has been plenty hard enough for me.

The music option is controlled by the SELECT switch. By the way, this music is some of the best I've heard on any game so far. The music option is available only on the disk version. I also appreciate the fact that you are not required to listen to 10 to 20 seconds of music before you can start the game. You start or restart the game by pressing START. Even after the game has begun, you may still change speeds or toggle the music using the appropriate key.

Once the game gets underway, you face the challenge of *Frogger*. The primary task is to hop frogs, one at a time, across a highway and a river into one of the five "homes" on the far side. On the display, this journey starts at the bottom of the screen and ends at the top. You control the movement of each frog with the joystick. Each frog is able to hop forward, backward, left, and right, but not at any of the 45-degree angles. To make it hop, you push the joystick in the desired direction. To make it hop again, you must return the stick to the neutral position and push it again in the desired direction.

Rest On The River Bank

The first obstacle is the highway. This involves crossing four lanes of traffic. To make this phase less than simple, the direction of traffic alternates with each lane, and the speed of each lane is different. Once you have made it past the highway, you can rest on the river bank before tackling the river. The river contains five "lanes." The first and fourth lanes contain turtles swimming upstream (i.e., to the left as you face the screen). The second, third, and fifth lanes contain logs which are floating downstream (i.e., to the right).

Naturally, these lanes move at different speeds. You cross the river by hopping on top of the turtles and logs to go from lane to lane. On the far side of the river, there is a wall with five little arches which represent the frogs' homes. You must hop directly from the last lane of logs into the arch to reach home.

The object of the game is to accumulate as many points as possible until you lose five frogs. The frogs may be lost in a number of different ways. The two most common ways: they are struck by a vehicle on the highway, or they fall into the river (swept away by the current, I assume). One feature that makes the river slightly more difficult to cross is that some of the turtles will submerge, taking your frog into the water with them. You also lose the frog if it misses the arch and hits the wall instead.

Finally, there is a time limit within which the frog must reach home. Each frog gets 120 counts which amounts to about 32 seconds. The "clock" appears as an orange bar at the bottom of the screen, and it shortens as the time runs out. When a frog is lost, a skull and crossbones appears briefly at the frog's last position. In addition to the "bleep" when the frog hops, there are appropriate sound effects when a frog is lost.

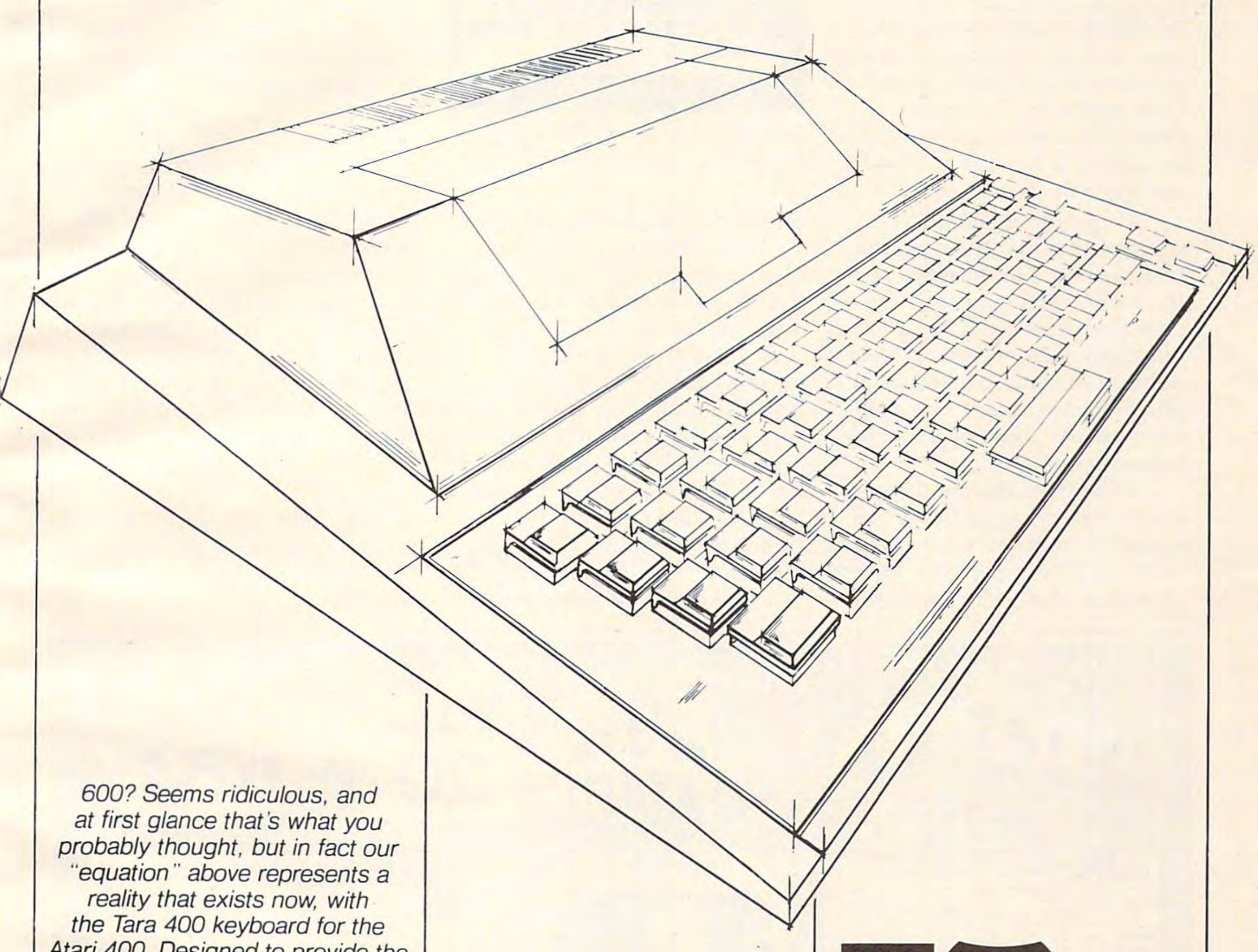
Points are accumulated in a number of ways. First, you receive 10 points for each forward jump your frog makes and 50 points for each frog that arrives home. Second, when a frog reaches home, you get 5 points for each count remaining on the time clock. In addition, there are a couple of ways to earn bonus points. From time to time an insect will appear in one of the unoccupied homes. If your frog can pounce on this insect, you receive 200 bonus points.

Alligators, Snakes, And Otters

A typical game consists of a sequence of rounds, once you can get past the first. A round is completed by maneuvering a frog into each of the five homes. There is a 1000-point bonus for completing a round, and you get to move on to the next round. Naturally, the level of difficulty increases for each successive round.

First of all, the traffic pattern on the highway changes: the amount of traffic increases, and the pattern requires more maneuvering to get across. The pattern of turtles and logs in the river also changes. Fewer logs appear in the third lane, and some of the logs in the last lane are replaced by alligators. You may hop on the backs of the alligators, but if you come too close to an alligator's mouth, the frog is eaten. Occasionally, an alligator will appear in one of the unoccupied homes for a brief period. You can wait for the alligator to leave or choose a dif-

400 + 400 = 600?



600? Seems ridiculous, and at first glance that's what you probably thought, but in fact our "equation" above represents a reality that exists now, with the Tara 400 keyboard for the Atari 400. Designed to provide the Atari 400 user with the hardware of tomorrow, today. Designed with an understanding of the essential superiority of a keyboard as a man–machine interface. Designed with the user in mind. For example, our keyboard does not attach to the 400 with a ribbon cable, but fits neatly into the original housing in 5 minutes, directly replacing the old membrane panel, and is styled to complement the lines of the computer itself. Sure, other keyboards have been sold, but who wants one that hangs off the computer, or whose keys fall off when you type on it? Our keys are actually gold-contact switches, offering increased reliability and performance, second to none. Coupled with the Tara 48K RAM expansion board, you can easily see how 400 + 400 = 600,* providing the user today with the hardware of tomorrow.

Why wait? This and many of the quality Tara products are waiting for you at your favorite dealer. Or call us for the Tara dealer nearest you. He'll be happy to show you how rudimentary it can all be with Tara.

Tara
Computer Products Inc.

Statler Building, 107 Delaware Ave.,
Suite 1610, Buffalo, N.Y. 14202 (716) 855-0133

2 Robert Speck Parkway, Suite 1540,
Mississauga, Ontario L42 1H8
(416) 273-6820

TARA PRODUCT LINE:
- Atari 400 Keyboard
- Atari 48K RAM
- Atari 32K RAM
- Apple 16K RAM

COMING SOON FROM TARA
- Apple 64K/128K RAM
- IBM 256K RAM
- Atari 64K RAM

*600 – The Atari redesigned full keyboard version of the Atari 400.

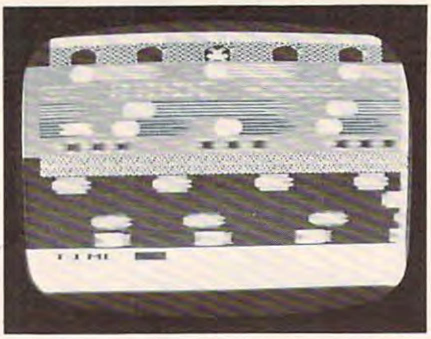
Atari and Atari 400 are registered trademarks of Warner Commmunications.

ferent home.

When you reach the third round, the difficulty is increased further by two additional predators: snakes and otters. The snakes will appear on the logs in the third lane of the river, and on the river bank between the river and the highway. If a snake appears, it will slither back and forth on the log. It is relatively safe to hop onto a log patrolled by a snake, since the snake will not chase the frog.

However, if you let the snake slither into the frog, the frog is eaten. On the river bank, the snake will make only one pass along the bank, but another may appear from either direction a short time later. An important point is that you can hop away from a snake, but you cannot hop over one.

The otters also are dangerously hungry, and can appear anywhere in the river, swimming between logs or between groups of turtles. The otters appear only on the disk version of the game.



Speeding cars, turtles, and logs whiz by in the official Atari version of Frogger.

After playing *Frogger* for quite a few hours, I would have to rate it as among the best games available for the Atari. It has very good graphics with lots of motion. In spite of all the motion, there is no noticeable jitter. Joystick response is very quick and quite sensitive. At first I found the joystick too sensitive. It was very easy to hop once too often or hop in the wrong direction. However, as your skill increases, this sensitivity becomes more and more valuable.

One of the best features of the game is the rate at which the level of difficulty increases with each round. The increased difficulty noticeably adds to the challenge, but is not so great as to cause undue frustration while trying to reach the higher levels. With the music thrown in for good measure, the game is a sure winner.

The Atari version of *Frogger* is very close to the real arcade game. There are only a few differences. First, there is twice the number of counts in the time limit to get the frog home. Along with this change, you receive only five bonus points for each count remaining on the clock. The only other significant difference is that the Atari version allows only one player.

Frogger
Sierra On-Line Inc.
36575 Mudge Ranch Road
Coarsegold, CA 93614
$34.95

©

VIC Rabbit: A High-Speed Cassette Interface

Roger N. Trendowski

With a very unassuming name, the VIC Rabbit may be one of the most useful peripherals you can add to your VIC-20 cassette system. What is a VIC Rabbit? It's a Read Only Memory (ROM) cartridge, manufactured by Eastern House Software, that plugs into the VIC expansion port.

VIC Rabbit adds 12 new commands to VIC BASIC. Of these, the new SAVE, LOAD, and VERIFY commands are the most impressive. (More about these commands later.) The Rabbit consists of two ROM sockets mounted on a circuit board; a ROM is mounted in one socket and the other is empty. Eastern House has plans to develop additional ROMs; possibly other utility programs or a word processor.

Also mounted on the circuit board are some capacitors, a controller chip, and at the rear, a female 44-pin connector. This expansion connector allows you to add additional memory, another utility cartridge, or even an expander motherboard with multiple slots. In other words, the Rabbit won't take over your expansion port.

SAVE, LOAD, And VERIFY

To activate the Rabbit, type the following command: SYS 7*4096. The manufacturer's name will be displayed, followed by READY.

Two SAVE commands, a LOAD command, and a VERIFY command are the main capabilities of the Rabbit. One of the SAVE commands, *S, is for saving programs with a six-second tape leader. The other SAVE command, *SS, provides only a three-second leader. The general format of the SAVE commands:

```
*SS"name",1,xxxx,yyyy
*SS"name"
*S
```

The name and device number are optional arguments, as they are with the normal SAVE command. The optional xxxx and yyyy fields are hexadecimal addresses which allow you to designate where in Random Access Memory (RAM) you want the program when reloading it.

Loading is done with a similar format:

```
*L"name",1,xxxx
*L"name"
*L
```

Normally, a program is loaded into memory at the same location from where it was previously saved. If you load a program with *L"name",1,f00, for example, the program will be loaded at the starting address of $0f00 (3840 decimal). After the program is loaded, the program name, program length (hex), starting address, and ending address are displayed in reverse video.

The format for verifying a program is:

```
*V"name"
*V
```

This VERIFY command is a different technique than is normal for the VIC-20. To guarantee a good recording, the standard VIC VERIFY command reads a program from tape and compares it (byte-for-byte) with the program stored in RAM. With the VIC Rabbit, a checksum value is calculated as the program is read from the tape. (The checksum is the cumulative total of the value of each byte in the program.) Next, the computed checksum is compared with a checksum which was saved on tape at the end of the program. Because Rabbit never compares the tape version with a copy in RAM, you don't actually need the program in RAM to verify it.

Saves Five Times Faster

The Rabbit performs a short SAVE five times quicker than the normal VIC SAVE. For example, for a 3K byte program, the SAVE command process takes one minute and nine seconds; the Rabbit, 15 seconds. For a 16K program, SAVE takes five minutes; the Rabbit takes only one minute.

The *S commands takes three seconds longer than *SS, since a longer leader is put on the tape. The *S command can be used to move past the plastic leader found at the beginning of most cassette tapes. Both the LOAD and VERIFY Rabbit commands take the same length of time as the SAVE command.

I did not actually test for error-rates on the Rabbit; however, I did record the number of bytes and the number of SAVEs I performed over a four-hour period of testing. I experienced no loading errors with 75 SAVEs and LOADs (involving about 200K).

There are three limitations to the Rabbit. First, it obviously cannot load a program which was saved in regular VIC tape format. Second, multiple commands cannot be used, e.g., *S:*S. Third, the Rabbit does not transfer data files to cassette tape. The PRINT# command, which VIC Rabbit does not affect, normally does this in the VIC-20.

Other Features

***E,"name"** – This command loads a program and then automatically runs it.

***T,v,xxxx,yyyy** – This command performs RAM tests in the memory range xxxx to yyyy (hex). If v = 1 for Test 1, the Rabbit tests RAM chips for storage retention. If v = 2 for Test 2, the chips are tested for proper selection operation. If an error is found, the bad memory address is printed out, along with the test pattern and error pattern numbers.

***Hxxxx** and ***Dxxxx** – These commands convert a hex number to decimal and decimal to hex. They are especially useful when working with the hex starting address and number of program bytes which are printed out with the load and verify commands.

***Gxxxx** – Go to machine language program at hex address xxxx.

* – This is one way to get back to the normal VIC screen. Pressing the RUN/STOP and RESTORE keys also works.

***Z** – This command switches the VIC to the graphics character set, or back to the alphanumeric character set. This command may be useful on other Commodore computers; however, on the VIC you need only to press the COMMODORE and SHIFT keys to accomplish the same thing.

***K** – Use this command to kill the Rabbit (disable the link). To reactivate the Rabbit link, you must type SYS 7*4096.

Configurations

VIC Rabbit can be used in a variety of configurations, with memory expansions, utility and graphics cartridges, and expansion motherboards. Rabbit uses address space in the third 8K RAM expansion block (from decimal 28672 to 32767). This leaves room for 27K of memory expansion. The Rabbit should be configured in series with an expansion motherboard if you intend to use one.

If you load the motherboard with both 8K + 16K, the Rabbit will isolate that portion of memory that overlaps into the address space that it uses. A block three memory expander should not be plugged in parallel with the Rabbit since that places two circuit controllers on the same VIC input lead. VIC will not know which device to take orders from.

According to Eastern House Software, block three RAM address space was used for the VIC Rabbit program so that the ROM area (decimal 32768 - 36869) could be reserved for other utility programs.

Documentation

An eight-page booklet is supplied with the VIC Rabbit. Except for the first page, it is easy to read. Page 1 is somewhat confusing because of its discussion of Rabbit interfaces with Commodore 64, PET Model 2001 and 4001, and CBM 8032. A separate VIC-20 instruction book will be available soon, according to the manufacturer.

The VIC Rabbit is an impressive product which should be extremely useful for BASIC and machine language programmers. It is built with expansion in mind and is both innovative and inexpensive.

VIC Rabbit
Eastern House Software
3239 Linda Drive
Winston-Salem, NC 27106
$39.95

©

VIC20™ PERSONAL COMPUTER

TOP 10 SALE!! ARCADE GAMES

(VIC-20 TAPE PROGRAMS)

Rank	Name	List	Sale
1.	Super Paratrooper (Fantastic)	$24.95	$19.95
2.	Exterminator-Plus (Better than Centipede)	$24.95	$19.95
3.	Cricket (Better than Frogger)	$24.95	$19.95
4.	3-D Hackman (3-Dimensional)	$24.95	$19.95
5.	Snackman (Better than Packman)	$19.95	$15.95
6.	Bug Blast (Creepy)	$19.95	$16.95
7.	Anti Matter Splatter (Nuclear Disaster)	$24.95	$19.95
8.	Bombs Away (Great)	$18.95	$15.95
9.	3-D Maze-Escape	$16.95	$14.95
10.	Krazy Kong	$16.95	$14.95

BUY ANY FOUR — DEDUCT 10% MORE

VIC-20 ACCESSORY SALE!!

1. **TRACTION-FRICTION LINE PRINTER** This new COM-STAR deluxe printer, prints 8½" x 11" full size letter quality single, roll or fan fold paper, labels and etc. Impact dot matrix bi-directional 40, 66, 80, 132 columns. Includes interface cable that plugs direct into the VIC-20 computer, no other costly interface is needed. List $599.00 — Sale $399.00.

2. **UP TO 60K EXPANSION MODULE** Aero Space designed—6 slot—add up to 6 cartridges—switch select any program. Start and stop any program with reset button—not necessary to remove cartridges or turn off computer, saves time, television and computer (one year warranty) List $149.00 — Sale $79.00.

- **10 DAY FREE TRIAL**
- **WE HAVE THE LOWEST PRICES**
- **ONE DAY DELIVERY EXPRESS MAIL**
- **FREE CATALOGS**
- **WE LOVE OUR CUSTOMERS!**

PROTECTO ENTERPRIZES (FACTORY-DIRECT)

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

YOU ARE GOING TO SMILE!

--When you see over 2000 books, programs, and accessories we carry for All Major Brands in our Computer 'SOURCE BOOK'. From introductory level to professional, we cover it all. Listed Below is just a sample of what we carry.
DEALER INQUIRIES WELCOME

JMC

NEW

NEW Vic 20™ GAMES

Direct from England, JMC is proud to to announce that we have been appointed exclusive U.S. importer for the following VIC-20® games. . .

MINEFIELD.A game for the 5K Vic 20. For Ages 6 and over. Your task is to drive an ambulance around a battle ground and pick up injured soldiers--racing the clock,and avoiding touching off the buried mines. You can adjust the complexity of the game by choosing to play against 25 to 100 mines.
4002-000001 $15.98

MAZE OF DEATH. For 5K Vic-20. For ages 8 and over. You have to travel across the screen thru a maze of walls blocking your path. Unseen hobgobblens and monsters may capture you before you reach your safe haven-on the other hand you may find buried treasure to add to your fortune. Full details scroll on the screen when you start game. 7 skill levels and 10 sensitivity levels to choose from.
4002-000002 $15.98

SPLOTTER© For 5K Vic-20
For ages 8 and over. You control a worm type animal that moves over the screen to gobble up the ugle spots--a race against time. If you bump into a wall, or turn back on yourself to quickly--you LOSE. You choose skill level from #1 (hardest) to #20 (easiest). If you really want to drive someone crazy, buy him this game!
4002-000003 $15.98

GUZZLER© For 5K Vic-20
For ages 8 and over. You must pilot you space shuttle thru a maze of asteroids to your mothership. But if you are not careful you will awaken the Guzzler©'s space bouys. You have to reach your mothership before running out of fuel. GOOD LUCK.
4002-000004 $15.98

DELUX 6-PAK© For 5K Vic-20.
Here is your chance to buy **6 games** on one cassette for the price of one game! You get all six of the following games for ONE low price!
War. You are defending against waves of enemy tanks. 3 levels of play...we bet you can't win at level 3!
Smashout. This version of bricks game tests skill,not reaction time,plan ahead!
Blackjack Popular card game '21',if you practice long enough maybe you can plan a system to beat Las Vegas!
Logic: You have 12 tries to break the code of 5 balls in random color sequence.
Pickupgame. You have to remove logs from the pile without disturbing any other logs.
Alarm Clock: Use the Computer as a REAL 24 hour alarm clock, bet you can't ignore its call when the alarm goes off!
4003-000001 $19.98

ANT RAIDERS© For 5K Vic-20
Ages 8 and up. An army of ants is marching its way across the land leading to your home. The land is littered with rocks which make the ants change direction often. You must shoot the ants before one gets to your door. If you shoot a rock it will divide into more rocks. If you trap all the ants, then more will come until you trap all the group, then more will come until...well you get the idea...have FUN!
4003-000003 $15.98

MORE GAMES COMING! We are working on more Vic-20™ games which we will be importing in the near future...watch our ads for details!

VIC-20® BOOKS

UNDERSTANDING YOUR VIC, VOL. 1 Beginners guide to programming on the VIC-20.
4840-000003 148 Pages $11.95
4840-000004 Cassette Suppl. $ 7.95

COMPUTE!'s FIRST BOOK OF VIC Collection of the best articles from Compute! Magazine.
4105-000007 $12.95

STARTING WITH 'BASIC' ON THE VIC-20® Helps novices learn all the capabilities of the versatile VIC-20.
4725-007070 128 Pages $12.95

KIDS AND THE VIC Super beginner's book from publisher of *Kids & Apple* Series. Makes computer experts out of kids (and their unsuspecting parents!).
4560-000056 $19.95

VIC-20 PROGRAMMER'S REFERENCE GUIDE Complete BASIC vocab guide, machine language programming, tips, and more!
4100-000110 $16.95

JMC Computer "SOURCE BOOK"© Over 2,000 books, programs and accessories for virtually every brand of micro computer made!
0001-198207 72 Page Catalog **$ 2.00 Ppd**

NEW

COMMODORE 64® BOOKS

THE ELEMENTARY COMMODORE 64® The easy way to understanding the *NEW* **C-64** from the author of *The Elementary Apple*. Good for new computer enthusiasts as well as for the new owner of the 64.
4560-000034 $14.95

The CBM-64® PROGRAMMER'S REFERENCE GUIDE *NEW* Everything you need to know to get started programming your C-64!
4760-022056 $19.95

ATARI 400/800® BOOKS

YOUR ATARI® COMPUTER: A GUIDE TO ATARI 400/800® PERSONAL COMPUTERS Invaluable reference. Complete "How-To" guide. Covers peripherals to programming.
4665-000065 $16.95

KIDS AND THE ATARI® Super begginer's text in the famous *"KIDS AND. . ." series.*
4560-000055 *NEW* **$19.95**

CONFUSED? No one book may offer all the answers. We suggest buying more than one book on a subject to get as many different views and opinions as possible. Books are inexpensive if you consider the time and frustrations they can avert!

NEW

THE BASIC BOOK: A Cross Referenced Guide to the BASIC Language. Convenient guide to different implementations of BASIC language by Apple®, Atari®, Texas Instruments®, Commodore®, IBM®, and Radio Shack®. Commands and syntax for each version are covered with examples of the more difficult concepts. Commands, statements and functions listed alphabetically. A special section devoted to graphics statements included. A definite must for your library.
4525-027959 *NEW* **$ 7.95**

DEPARTMENT: Z-T
1025 INDUSTRIAL DRIVE
BENSENVILLE, IL 60106-1297

QTY	NUMBER	DESCRIPTION OF ITEM	COST	TOTAL
			TOTAL	

(Orders Under $25 Please Add $2.50 Post & Handling. Illinois Residents Please Add 5% Sales Tax. ALL CATALOGS SHIPPED POST PAID.)

PAYMENT ENCLOSED: □ CASH □ CHECK □ MONEY ORDER

PLEASE CHARGE TO MY □ MASTERCARD □ VISA (Min Chg. $25)

CARD NUMBER ____________________

EXPIRES ____________ INTRBNK # ____________

SHIP TO: ____________________

STREET ADDR: ____________________

CITY ____________ STATE ______ ZIP ______

JMC

PRICES AND AVAILABILITY SUBJECT TO CHANGE

Alspa Computer, Inc.

The price-performance leader. Includes Z80A, 1 or 2 full 8" drives (double density, double sided), 3 serial and 1 parallel port, and winchester port. Prices start at less than $2000. DEALER and OEM inquiries invited.

SPECIALS on INTREGATED CIRCUITS

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

Anchor Automation Signalman Modems

FREE SOURCE MEMBERSHIP WITH SIGNALMAN

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

Mark I RS232
Mark II for Atari 850
Mark IV for CBM/PET with software
Mark V for Osborne (software available)
Mark VI for IBM Personal Computer
Mark VII Auto Dial/Auto Answer
Mark VIII Bell 212 Auto Dial/Answer

DC HAYES Smartmodem	**229**
RS232 MODEM — Accoustic	**119**
RS232 MODEM — CCITT frequencies	**175**

We carry Apple II+ from Bell & Howell

16K RAM Card for Apple	**65**
Apple LOGO	150
Video Recorder Interface	545
Super Serial Card	149
Thunderclock Plus	119
Z80 Softcard and CP/M	295
Parallel Printer Interface/Cable	80
Grappler Interface	139
TG Products Joystick for Apple	**48**
TG Paddles	32
DC Hayes Micromodem II	299
Videx 80 Column Card	259
fullFORTH+ for Apple (fig-Forth)	85
Silentype Printer and Card	310
Graphics Tablet and Card	645
Apple PASCAL Language	195
Apple FORTRAN	160
We stock EDUWARE Software	
GENIS I Courseware Development System	145
Unicom Grade Reporting or School Inventory	250
Executive Briefing System with fonts	225
Apple Dumpling (Microtek) Printer Interface	**115**
Apple Dumpling with 16K Buffer	**160**
PIE Writer Word Processor	**120**

commodore

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

PETSCAN $245 base price

Allows you to connect up to 35 CBM/PET Computers to shared disk drives and printers. Completely transparent to the user. Perfect for schools or multiple word processing configurations. Base configuration supports 2 computers. Additional computer hookups $100 each.

Commodore COMMUNICATES!

COMPACK $129

Intelligent Terminal Package includes:
ACIA hardware based interface; DB25 Cable and STCP Software with remote telemetry, transfer to/from disk, printer output, XON-XOFF control, user program control, and status line.

VE-2 IEEE to Parallel Interface 119

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

VIC 20	189	VIC Sargon II Chess	32
VIC Printer	335	VIC GORF	32
VIC 3K RAM	32	Meteor Run (UMI)	39
VIC 8K RAM	53	VIC Radar Ratrace	24
VIC 16K RAM	99	Amok (UMI)	20
VIC Disk Drive	395	Snakman	15
VIC Pinball	32	Rubik's Cube	13
VIC Omega Race	32	Programmers Reference	15
Spiders of Mars (UMI)	39	Renaissance (UMI)	39
VIC Draw Poker	24	VIC Superslot	23

VICTORY Software for VIC

Street Sweepers	12	Maze in 3-D	12
Night Rider	11	Cosmic Debris	12
Treasures of Bat Cave	12	Grave Robbers Advent	11
Games Pack I	12	Games Pack II	12
Victory Casino	8	Adventure Park I	12
Adventure Pack II	12	Trek	11

TNW 488/103 with DAA	450
Compute!'s First Book of PET/CBM	11
POWER ROM Utilities for PET/CBM	78
WordPro 3+ - 32K CBM, disk, printer	195
WordPro 4+ - 8032, disk, printer	300
SPELLMASTER spelling checker for WordPro	**170**
COPY-WRITER Professional Word Processor	159
VISICALC for PET, ATARI, or Apple	190
PETRAX PET to Epson Graphics Software	**35**
SM-KIT enhanced PET/CBM ROM Utilities	**40**
Programmers Toolkit - PET ROM Utilities	35
PET Spacemaker II ROM Switch	36
2 Meter PET to IEEE or IEEE to IEEE Cable	40
Dust Cover for PET, CBM, 4040, or 8050	8
VIC or C64 Parallel Printer Interface	85
CmC IEEE-RS232 Printer Interface — PET	120
SADI Intelligent IEEE-RS232 or parallel	235
Library of PET Subroutines	12
Programming the PET/CBM (Compute!) — R. West	**20**
Compute! First Book of VIC	11
Whole PET Catalog (Midnight Gazette)	**8**
Color Chart Video Board for PET	125
PET Fun and Games (Cursor)	11

REVERSAL (Spracklen) Apple or Atari	25
SARGON II — Apple or TRS-80	26
Apple II User's Guide (Osborne)	12
Introduction to Pascal (Sybex)	13
Pascal Handbook (Sybex)	16
Musical Applications of Micros (Chamberlin)	20
Starting FORTH	14
Discover FORTH	12
User Guide to the Unix System	13
6502 Assembly Language Subroutines	11
PET Fun and Games	9
KAMIKAZE (Hoyden Software-Apple)	28

DISK SPECIALS

SCOTCH (3M) 5"	10/ 2.30	50/ 2.10	100/ 2.05
SCOTCH (3M) 8"	10/ 2.45	50/ 2.20	100/ 2.15

We stock VERBATIM DISKS

Write for Dealer and OEM prices.

BASF 5" or 8"	10/ 2.00	20/ 1.95	100/ 1.85
Wabash 5"	10/ 1.80	50/ 1.75	100/ 1.70
Wabash 8"	10/ 2.00	50/ 1.95	100/ 1.90

We stock MAXELL DISKS

Write for dealer and OEM prices.

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

CASSETTES—AGFA PE-611 PREMIUM

High output, low noise, 5 screw housings.

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

SPECIALS

Timex/Sinclair Computer	79
Zenith ZVM-121 Green Phosphor Monitor	109
VOICE BOX Speech Synthesizer (Apple or Atari)	
BMC Green and Color Monitors	
Many printers available (Star, Brother, OKI, etc.)	
We Stock AMDEK Monitors	
Watanabe Intelligent Plotter 1095	6-pen 1395
Staticide anti-static spray	6
dBASE II	390
Panasonic TR-120M1P 12" Monitor (20 MHz)	149
Panasonic CT-160 Dual Mode Color Monitor	285
Franklin Computers - special system price	
Hewlett Packard Calculators	

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

ALL BOOK and SOFTWARE PRICES DISCOUNTED

A P Products	**15% OFF**
Synertek SYM-1 Microcomputer	**SALE 189**
KTM-2/80 Synertek Video and Keyboard	349
KTM-3/80 Synertek Tubeless Terminal	385

ZENITH data systems

Z90-80 64K	1995
Z90-82 64K, 1 double dens. drive	2245
Z37 1.3 Megabyte Dual Drive	1355
Z19 Video Terminal (VT-52 compatible)	695
ZT-1 Intelligent Communications Terminal	**479**
Z100 16-bit/8-bit System	**CALL**

ATARI SPECIALS

800 Computer	669	Microsoft BASIC	72
400—16K	269	**MISSILE COMMAND**	**29**
810 Disk Drive	440	**ASTEROIDS**	**29**
825 Printer	625	**STAR RAIDERS**	**34**
850 Interface	170	Space Invaders	29
Inside Atari DOS	18	Music Composer	35
Joysticks or Paddles	19	Caverns of Mars	33
16K RAM (Microtek)	69	**PAC-MAN**	**36**
32K RAM (Microtek)	99	**CENTIPEDE**	**36**
Pilot	65	First Book of Atari	11
Super Breakout	29	**Anchor Modem—Atari**	**85**
APX Software	Call	Other Atari products	Call

252 Bethlehem Pike
Colmar, PA 18915

215-822-7727

A B Computers

WRITE FOR CATALOG

Add $1.25 per order for shipping. We pay balance of UPS surface charges on all prepaid orders. Prices listed are on cash discount basis. Regular prices slightly higher. Prices subject to change.

KMMM Pascal for PET/CBM $85

A subset of standard Pascal with extensions.

- Machine language Pascal Source Editor with cursor oriented window mode
- Machine Language P-Code Compiler
- P-Code to machine language translator for optimized object code
- Run-time package
- Floating point capability
- User manual and sample programs

Requires 32K Please specify configuration.

EARL for PET (disk file based) $65

Editor, Assembler, Relocater, Linker

Generates relocatable object code using MOS Technology mnemonics. Disk file input (can edit files larger than memory). Links ,multiple object programs as one memory load. Listing output to screen or printer. Enhanced editor operates in both command mode and cursor oriented "window" mode.

RAM/ROM for PET/CBM

4K or 8K bytes of soft ROM with optional battery backup.

RAM/ROM is compatible with any large keyboard machine. Plugs into one of the ROM sockets above screen memory to give you switch selected write protectable RAM.

Use RAM/ROM as a software development tool to store data or machine code beyond the normal BASIC range. Use RAM/ROM TO LOAD A ROM image where you have possible conflicts with more than one ROM requiring the same socket. Possible applications include machine language sort (such as SUPERSORT), universal wedge, Extramon, etc.

RAM/ROM – 4K	$75
RAM/ROM – 8K	90
Battery Backup Option	20

SUBSORT by James Strasma $35

Subsort is an excellent general purpose machine language sort routine for PET/CBM computers. Sorts both one and two dimensioned arrays at lightning speed in either ascending or descending order. Other fields can be subsorted when a match is found, and fields need not be in any special order. Sort arrays may be specified by name, and fields are random length. Allows sorting by bit to provide 8 categories per byte. The routine works with all PET BASICs, adjusts to any memory size, and can co-exist with other programs in high memory.

SuperGraphics 2.0

NEW Version with TURTLE GRAPHICS

SuperGraphics, by John Fluharty, provides a 4k machine language extension which adds 35 full featured commands to Commodore BASIC to allow fast and easy plotting and manipulation of graphics on the PET/CBM video display, as well as SOUND Commands. Animations which previously were too slow or impossible without machine language subroutines now can be programmed directly in BASIC. Move blocks (or rocketships, etc.), or entire areas of the screen with a single, easy to use BASIC command. Scroll any portion of the screen up, down, left, or right. Turn on or off any of the 4000 (8000 on 8032) screen pixels with a single BASIC command. In high resolution mode, draw vertical, horizontal, and diagonal lines. Draw a box, fill a box, and move it around on the screen with easy to use BASIC commands. Plot curves using either rectangular or polar co-ordinates (great for Algebra, Geometry and Trig classes.)

The SOUND commands allow you to initiate a note or series of notes (or even several songs) from BASIC, and then play them in the background mode without interfering with your BASIC program. This allows your program to run at full speed with simultaneous graphics and music.

Seven new TURTLE commands open up a whole new dimension in graphics. Place the TURTLE anywhere on the screen, set his DIRECTION, turn him LEFT or RIGHT, move him FORWARD, raise or lower his plotting pen, even flip the pen over to erase. Turtle commands use angles measured in degrees, not radians, so even elementary school children can create fantastic graphic displays.

Specify machine model (and size), ROM type (BASIC 3 or 4)

SuperGraphics in ROM $45

Volume discounts available on ROM version for schools.

FLEX-FILE

for PET/CBM Computers

NEW VERSION II

FLEX-FILE is a set of flexible, friendly programs to allow you to set up and maintain a data base. Includes versatile Report Writer and Mail Label routines, and documentation for programmers to use Data Base routines as part of other programs.

RANDOM ACCESS DATA BASE

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

MAILING LABELS

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

REPORT WRITER

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

FLEX-FILE II by Michael Riley $110

Please specify equipment configuration when ordering.

DISK I.C.U. $40

Intensive Care Unit by L.C. Cargile

COMPLETE DISK RECOVERY SYSTEM FOR CBM DRIVES

- edit disk blocks with ease
- duplicate disks, skipping over bad blocks
- complete diagnostic facilities
- un-scratch scratched files
- check and correct scrambled files
- recover improperly closed files
- extensive treatment of relative files
- optional output to IEEE488 printer
- comprehensive user manual (an excellent tutorial on disk operation and theory).

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

PROGRAM YOUR OWN EPROMS $75

Branding Iron EPROM Programmer for PET/CBM software for all ROM versions. Includes all hardware and software to program or copy 2716 and 2532 EPROMs.

PORTMAKER DUAL RS232 SERIAL PORT $63

Two ports with full bipolar RS232 buffering. Baud rates from 300 to 4800. For PET/CBM, AIM, SYM.

Hunter-Killer – Commodore 64 15

Authentic naval warfare game (complete with sonar)

CBM Software

Petspeed BASIC Compiler	225
Integer BASIC Compiler	110
CMAR Record Handler	110
UCSD Pascal (without board)	135
Wordcraft 80 or 8096	265
BPI Accounting Modules	280
Professional Tax Prep System	575
Intelligent Terminal Emulator	25
ASERT Data Base	375
Dow Jones Portfolio Management	110
Assembler Development	80

FORTH for PET

BY L. C. Cargile and Michael Riley $50

Features include:

- full FIG FORTH model.
- all FORTH 79 STANDARD extensions.
- structured 6502 Assembler with nested decision making macros.
- full screen editing (same as when programming in BASIC).
- auto repeat key.
- sample programs.
- standard size screens (16 lines by 64 characters).
- 150 screens per diskette on 4040, 480 screens on 8050.
- ability to read and write BASIC sequential files.
- introductory manual.
- reference manual.

Runs on any 16K or 32K PET/CBM (including 8032) with ROM 3 or 4, and CBM disk drive. Please specify configuration when ordering.

Metacompiler for FORTH $30

simple metacompiler for creating compacted object code which can be executed independently (without the FORTH system).

PaperMate 60 COMMAND WORD PROCESSOR

by Michael Riley

Paper-Mate is a full-featured word processor for CBM/PET by Michael Riley. Paper-Mate incorporates 60 commands to give you full screen editing with graphics for all 16K or 32K machines (including 8032), all printers, and disk or tape drives. Many additional features are available (including most capabilities of Professional Software's WordPro 3).

For writing text, Paper-Mate has a definable keyboard so you can use with either Business or Graphics machines. Shift lock on letters only, or use keyboard shift lock. All keys repeat.

Paper-Mate text editing includes floating cursor, scroll up or down, page forward or back, and repeating insert and delete keys. Text block handling includes transfer, delete, append, save, load, and insert.

All formatting commands are imbedded in text for complete control. Commands include margin control and release, column adjust, 9 tab settings, variable line spacing, justify text, center text, and auto print form letter (variable block). Files can be linked so that one command prints an entire manuscript. Auto page, page headers, page numbers, pause at end of page, and hyphenation pauses are included.

Unlike most word processors, CBM graphics as well as text can be used. Paper-Mate can send any ASCII code over any secondary address to any printer.

Paper-Mate functions with all CBM/PET machines with at least 16K, with any type of printer, and with either cassette or disk.

To order Paper-Mate, please specify machine and ROM type.

Paper-Mate (disk or tape) for PET, CBM, VIC, C64 **$40**

SM-KIT for PET/CBM $40

Enhanced ROM based utilities for BASIC 4. Includes both programming aids and disk handling commands.

BASIC INTERPRETER for CBM 8096 $200

A full interpreter implementation to automatically take advantage of the extra memory available with 8096.

PEDISK II Systems from cgrs Microtech available.

FILEX IBM 3741/2 Data Exchange Software available.

JINSAM Data Base Management System for CBM.

Comprehensive version available for most configurations.

COPY-WRITER Word Processor for PET/CBM. $159

Works like expensive word processors, plus has added features like double column printing and shorthand generator.

CASH MANAGEMENT SYSTEM $45

Easy to use disk system. Keeps track of cash disbursements, cash receipts, cash transfers, expenses for up to 50 categories.

252 Bethlehem Pike
Colmar, PA 18915

215-822-7727

A B Computers

WRITE FOR CATALOG

Add $1.25 per order for shipping. We pay balance of UPS surface charges on all prepaid orders. Prices listed are on cash discount basis. Regular prices slightly higher. Prices subject to change.

Mathematics Action Games For TI

C. Regena

Parents and teachers – can you imagine a situation where you do *not* have to nag your students to practice their math? In our high tech society what could be more motivating than computer games? Scott, Foresman and Company has developed three command modules for the Texas Instruments TI-99/4A computer that combine color, graphics, animation, and music with educational concepts for some fun *Mathematics Action Games.*

Each module consists of two major games, and each game has three levels of difficulty, so the modules are versatile enough for a wide range of students. You may choose a one-player or a two-player game. The series supplements any major basal mathematics program. Each module is packaged in a durable vinyl album with a *Teacher's Guide*. The *Teacher's Guide* includes reproducible worksheets and record sheets.

Hard, Harder, Hardest

The modules are an intriguing way to practice fundamental math skills. A student interacts with the game after a correct response. If the student answers incorrectly, the correct response is supplied. There is a time element, so the more quickly the student answers, the higher the score will be (or more jumps or more bowling pins, etc.). Scott, Foresman produces the Mathematics Courseware Series to teach and to give tutorial and remedial help. The *Mathematics Action Games* provide the practice.

Module A is for kindergarten through third grade and has difficulty levels of Hard, Harder, and Hardest. *Frog Jump* at the Hard level is a game involving identifying one greater than or one less than a given number. The Harder level employs the concept of the next number in a series counting by twos, fives, and tens. The Hardest level involves order multiples of 10, 100, and 1000. A sample problem is "Give 100 more than 8396." Depending on your answer, your frog jumps a certain number of lilly pads.

The second game in Module A is *Picture Parts,* which gives practice in the basic mathematical operations. For the Hard section, you give answers to basic addition and subtraction questions such as 9+3=?. In the Harder level, you give the missing number in basic addition and subtraction equations, such as 12 - ? = 5. The Hardest level requires answers to basic multiplication questions.

Module B is designed for grades three through six and provides practice with multiplication and division. *Pyramid Puzzler* is the game for multiplication. The Standard level involves giving missing multiplication factors. The Advanced level involves multiplying by multiples of 10 and 100. A sample problem is to multiply 7 x 400.

Ready to practice division? Try the game of *Star Maze.* All the problems are written in standard division form (long division). The Standard level requests answers to basic division problems. The problems for the Advanced level involve dividing by a one-digit divisor to get a one-digit quotient with a remainder. The Master level problems ask you to divide a three-digit dividend by a one-digit divisor.

From Amateur To Champion

Module C of the *Math Action Games,* one of my favorite modules, is for 6th, 7th, and 8th grades. The graphics for *Number Bowling* are really good. To get a strike, you must give the correct response almost immediately. The longer it takes to answer, the fewer pins you'll hit – and an incorrect answer is a gutter ball. *Number Bowling* has problems involving decimals and fractions. The Amateur level has two types of problems. One type asks you to compare and order decimals (Which is greater? 3.0254 3.3025). The second type of problem asks you to write decimals, such as "Give as a decimal: fifty-one ten-thousandths." The Pro level requires you to write a fraction given a mixed number or to write a mixed number given an improper fraction. The Champion level involves writing a decimal equivalent of a given fraction.

Space Journey gives practice with decimals and percents and at the same time satisfies anyone's urge for a space game. Using the arrow keys, you can land on asteroids or planets, answer a certain number of questions, and try to get to your destination as soon as possible. The Amateur level asks you to give decimal equivalents for percents greater than one and less than 100. The Pro level involves writing decimal equivalents for percents greater than 100 or less than one (Example: Give as a decimal .7%). The Champion level requires you to write percent equivalents for decimals, fractions, or whole numbers.

The *Math Action Games* are highly motivational, and, in the competitive formats, encourage quick thinking. Students will enjoy playing the games over and over to try to improve scores. It's practice at basic math concepts disguised as fun.

I highly recommend these modules for all classrooms up to 8th grade and for families with children under the age of 14.

Mathematics Action Games
Scott, Foresman and Company
1900 East Lake Avenue
Glenview, IL 60025
$75.95 per module

©

Facemaker And Story Machine For Apple, Atari And CBM

Sheila Cory

Spinnaker Software Corp. has developed two terrific ideas into programs for young children. *Facemaker*, designed for children aged four to eight, is a program that allows you to create and animate funny faces. *Story Machine*, for children aged five to nine, allows you to write a story at the keyboard and see it animated. One of these terrific ideas was developed into a terrific program, while the other one falls a bit short.

Facemaker

Have you ever had a secret desire to design and animate your own cartoon? Have you been looking for software that can be used to introduce computer programming to children as young as four? Do you like a good chuckle now and then? If you answered "yes" to at least one of these questions, *Facemaker* deserves your attention. Although designed for four- to eight-year-olds, *Facemaker* appeals to older children and adults as well.

Facemaker has three major options available. First, it allows you to design a face and, second, you can determine how the face should be animated. The third option is an entertaining and memory-building game.

To design a face, you choose from a series of menus of mouths, eyes, ears, noses, and types of hair. There are eight possible choices for each of these features; this permits a large number of possible different faces. As each feature is chosen, it is added to the face, so the face takes form right before your eyes. The choice of features is varied and creative, and the method of selecting them from the various menus is simple enough for even young children to learn.

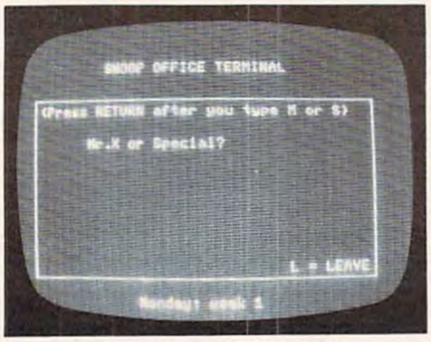
When you want to make another face, you replace the features, one by one, on the old face. It's almost like a metamorphosis taking place before your eyes. Some of the children who looked at this program felt they would like to be able to build all of their faces from scratch. This can't be done once the first face is created, unless you turn the computer off, take the diskette out, and begin all over again.

Once a face is designed, you can animate it. In animating the faces, you are essentially writing a program, a list of instructions for the face to follow. Choices here include wink, cry, smile, frown, tongue out, and ear wiggle. To make it easier for young children to use this program, each of these options can be specified with just its first letter.

A typical program might be WTWFCW-EEW. This would cause your face to wink, stick its tongue out, wink again, frown, cry, wink a third time, delay a moment, ear wiggle twice, and end with a wink. All of this can be accompanied by appropriate sounds. I've yet to see anyone, adult or child, animate a face and not giggle when the program is run.

The third option, or memory game, is also based on a face you've designed. If you choose to play the game without having designed a face first, you'll get *very* strange results, so be sure to play the game after you've built your face. In this game, the face you've designed will animate, and the object is for you to specify exactly what steps were involved in the animation. If you get it right, one more step is added to the animation the next time.

For example, the first time the face may wink and frown. If you respond with the correct sequence, you're rewarded with a smile and a wink, and a third action is added to the animation. This continues until a mistake is made. One criticism I have of this program is the way it handles a mistake – the tongue is stuck out, and unfriendly sounds announce your error. I would be sure to alert children to this before letting them use the program.



```
SNOOP OFFICE TERMINAL

(Press RETURN after you type M or S)

Mr.X or Special?

L = LEAVE

Monday: week 1
```

A typical screen common to both Snooper Troops #1 *and* Snooper Troops #2.

The screen layouts for this program are very pleasing, and not too complicated or crowded for use by young children. Documentation for *Facemaker* is minimal, but that's all that's needed since the program is very user-friendly and clearly explained.

Facemaker might be the ideal program to accomplish several different objectives. Introducing young children to programming by writing programs for the created characters would be an excellent way of exposing them to an important programming concept. This program would also be an ideal invitation to try a computer – for people of any age that have some anxiety about sitting down at the keyboard. I suspect that few people would turn down an opportunity to design their own animated cartoon. *Facemaker* also lends itself well to presentation to a group of people. Various people can contribute suggestions to the building of the face, and the animation can be a group effort.

Story Machine

Children love to make up stories. They also love to watch cartoons judging by the number of Saturday morning cartoons on television. A recognition of these things that are enjoyable to children is behind the development of *Story Machine*.

Using the words allowed in the program, you can write stories at the keyboard. As the story is typed in, *Story Machine* uses the top portion of the screen to illustrate the story. When a sentence involves action (such as "The dog goes to the store."), the program will actually animate the sentence.

Story Machine provides a dictionary of 45 words, divided into seven categories: (1) articles, such as *the, a,* and *an*; (2) adjectives, such as *this, that,* and *those*; (3) pronouns, such as *he, she,* and *it*; (4) nouns, such as *apple(s), dog(s),* and *girl(s)*; (5) possessive pronouns, such as *his, her,* and *its*; (6) verbs, such as *are, eat(s)* and *run(s)*; and (7) prepositions, such as *at, by,* and *to.*

The documentation lists the following rules for generating sentences that are acceptable in the program:

1. Begin each sentence with an article, pronoun, or adjective.

2. Use present tense verbs (*run,* not *ran*).

3. A period or prepositional phrase must follow a verb (run *to a house*).

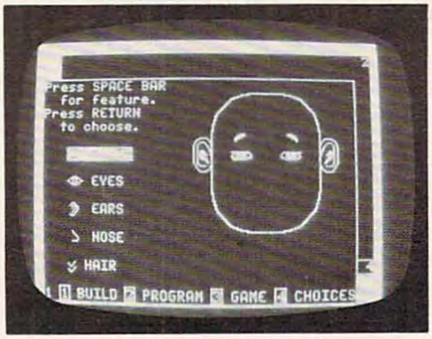
4. Pronouns can be used only for the last noun used as the subject of a sentence.

5. End each sentence with a period.

6. Only four actors (nouns) may appear on the screen at any one time.

Carefully following these rules, I designed a few simple stories to type in and animate. My first story was:

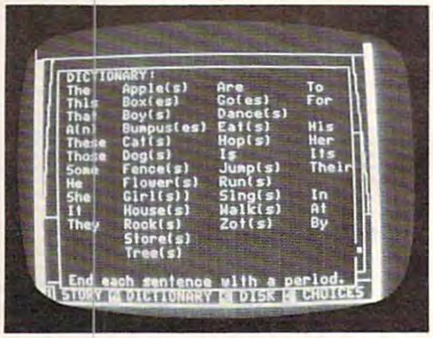
A BOY RUNS BY A BOX.
AN APPLE IS IN THE BOX.
THE BOY EATS THE APPLE.
THE BOY GOES IN THE BOX.



```
Press SPACE BAR
  for feature.
Press RETURN
  to choose.

  EYES
  EARS
  NOSE
  HAIR
1 BUILD 2 PROGRAM 3 GAME 4 CHOICES
```

Building a face with Facemaker.



```
DICTIONARY:
The    Apple(s)    Are       To
This   Box(es)     Go(es)    For
That   Boy(s)      Dance(s)
A(n)   Bumpus(es)  Eat(s)    His
These  Cat(s)      Hop(s)    Her
Those  Dog(s)      Is        Its
Some   Fence(s)    Jump(s)   Their
He     Flower(s)   Run(s)
She    Girl(s))    Sing(s)   In
It     House(s)    Walk(s)   At
They   Rock(s)     Zot(s)    By
       Store(s)
       Tree(s)

End each sentence with a period.
1 STORY 2 DICTIONARY 3 DISK 4 CHOICES
```

Words are chosen from the dictionary to spin a yarn with Story Machine.

Eagerly anticipating seeing my story animated, I began to type in the first sentence. Here is a running commentary of my experience.

I typed "A BOY" (picture of a boy appeared on screen), then "RUNS BY" (BY got a slash through it and was then erased). I guessed I couldn't do that, so I typed in "TO A BOX" (picture of a box appeared on screen, and the boy ran to it!). Now, on to the second sentence. I typed "AN APPLE" (apple appeared on screen) "IS IN THE BOX" (message IT'S FULL appeared on screen, BOX got a slash through it, and my whole sentence was automatically erased).

Oh well. I decided I'd try a slight modification and continued typing. "AN APPLE" (apple appeared on screen) "IS BY THE BOX" (message NO SPACE appeared on the screen and, again, my whole sentence was erased, but the apple stayed on the screen although there was now no sentence mentioning it).

I decided to stop trying to relate the apple to the box, and went on to the next sentence. I typed "THE BOY EATS THE APPLE" (message MUST BE CLOSER appeared on screen, and again my sentence was erased). Assuming the boy needed to be closer to the apple in order to eat it, I typed in "THE BOY GOES TO THE APPLE" (boy on screen moved toward the apple).

Next I typed in "THE BOY EATS THE APPLE" (the apple slowly disappeared, and the boy moved to the spot where the apple had been – the boy did not appear to actually eat the apple). And finally I was at my last sentence, so I typed in "THE BOY GOES IN" (IN got a slash through it, then erased), so I gave up and typed "TO THE BOX". My finished story looked like this:

A BOY RUNS TO A BOX.
THE BOY GOES TO THE APPLE.

THE BOY EATS THE APPLE.
THE BOY GOES TO THE BOX.

I felt quite frustrated. That's not at all what I had wanted my story to say.

Typing in my other two stories proved equally frustrating. The sentences that were acceptable to the program were very limited, with more being rejected than accepted.

Story Machine allows stories, once written, to be saved on a diskette. This option would allow children to go back and enjoy their old stories over and over. In a classroom situation, it would also allow them to be shared with others.

The idea for creating a program like *Story Machine* is excellent and educationally valid. Unfortunately, in its present form it could well be more frustrating than educational for children at the appropriate age to use it. To be useful, it would need to accept a much broader array of sentences – ideally any sentence that is correctly formed with the words available in the limited dictionary. Because the program is such a terrific idea, I hope the Spinnaker people put out another version that will accept any syntactically correct sentences.

Other Spinnaker Software

I also took a quick look at *Snooper Troops*, Case #1 and Case #2. These two adventure games, designed for people aged ten to adult, are based on two different who-done-it type mysteries. The adolescents in my neighborhood loved them, comparing them favorably to other adventure games they've spent time on. They can develop logical thinking skills to narrow in on the culprit.

Facemaker *($34.95)*
Story Machine *($34.95)*
Snooper Troops #1 and #2
($44.95 each)
Spinnaker Software Corp.
215 First Street
Cambridge, MA 02142

©

Graphics And Programming Utilities For Sinclair/Timex

Arthur B. Hunkins

Softsync, publishers of high-quality software for Sinclair/Timex, has recently released two useful utility packs for the 16K Sinclair line – *Programmers Toolkit* and *Graphics Kit*. The two "kits" are compatible; i.e., they are designed to be used together (though they need not be).

Both are in machine language, hidden away – protected – in high memory. *Programmers Toolkit* occupies the top 1K; *Graphics Kit* requires an additional 1K-plus. The packages are not user-relocatable, no doubt in part because allowing this option would have compromised security; these programs are *well* protected.

Graphics is a collection of 23 graphics routines; *Toolkit* includes eight utilities. The nature of the packages is evident from their titles. The individual routines are quite useful, provided one doesn't expect too much from them. They operate by calling USR, often after POKEing specific information into key memory locations. This procedure is somewhat unwieldy, requiring you to remember – or constantly refer to – many different memory locations (i.e., five-digit numbers). *Graphics Kit* has no ready reference guide to these locations; the documentation for *Toolkit* is better in this regard.

The Most Important Routine

The software is well-written throughout. Aside from its cumbersomeness, I find only one thing to criticize about *Toolkit*: it omits a repeat key routine. Perhaps a repeat function was not considered appropriate for the package, but it surely would have been useful. I can't think of any utility that would do more to facilitate programming on the Sinclair/Timex.

The most important routine in *Toolkit* is *Renumber*. This routine renumbers an entire program, with any starting line number and any increment (default values are 10). It handles GOSUB and GOTO; the only difficulty – and something of a nuisance – is that GOTOs and GOSUBs must be four-digit numbers. If they are not, the routine ignores them.

Perhaps next most useful are *Search And List* and *Search And Replace*. The former lists line numbers (not entire statements) where a designated character is found; the latter replaces all occurrences of one designated character with another. Equally handy is *Free*, which prints the number of bytes free at any given point.

Reverse and *Fill* affect a specified number of lines on the screen, starting with the topmost line. *Reverse* inverts the field of the area; *Fill* fills it with any specified character. There is no provision for reversing or filling partial lines, or implementing any other kind of screen partition.

Probably the most fascinating routine is *Hyper-Graphics Mode*, which changes the start address of the character table. A demo program illustrates the "exploded" characters that result; clearly *Hyper-Graphics* is an entry into the intriguing world of user-created character graphics. Unfortunately, no further software is offered; let us hope that Softsync soon releases additional tools.

Finally, *Wait* does a program "hold" until it senses input from the cassette player, whereupon the BASIC program is continued. *Toolkit*'s documentation includes a set of short programs that nicely demonstrates the various utilities.

A Wealth Of Useful Aids

It is difficult to select highlights from *Graphics Kit*. There are a wealth of useful routines here; most are well illustrated in the demo program included on tape. The demo is impressive, and would serve well as a repeating window display. Perhaps the best way to become familiar with *Graphics* is to study and work with this program. It is entirely in BASIC. Note at the outset that *Graphics Kit* has no high-resolution capability, nor does it permit you to create your own character set.

It does, however, allow you to define your own multi-character shapes, and to *Draw* or *Undraw* (erase) them anywhere on the screen, using PRINT AT. *Draw*, probably the classiest utility of the package, lets shapes be defined in REM statements, using the usual graphics characters and a system of "cursor controls" (directional arrows).

One useful feature that appears in the demo, but is not referred to elsewhere, is the quote character ("), which serves as a cursor blank (in place of the space). There is a difference; "background" shows through the cursor blank, while it does not for the space. The demo shows a worm marching ("crawling" would be stretching a point) along the screen, *behind* a cactus and *in front* of a background. Not spectacular, but not bad either.

A number of other routines also "do" and "undo" various features: *Foreground On, Foreground Off; Border, Unborder; Onscreen, Offscreen; Background On, Background Off. Foreground On* is especially interesting, as it "protects" everything currently displayed, causing graphics added later to "pass behind" the protected display. *Background On* works on a related principle; it creates a background of a specified character upon which everything *else* is projected – including *Foreground* and other characters.

Thus a simple three-dimensionality is created. *Background* can include one or both edit lines, if desired. In general, *Graphics* routines can use or exclude these lines at the user's option. Most routines optionally permit working with fewer than 20 lines.

Border places your choice of character around the outside of the screen. *Offscreen* blanks the screen to its normal background color (not black), while saving its contents. The documentation stresses that Fast is not used in this operation.

A multi-directional scroll capability is also most attractive. *Upscroll, Downscroll, Rightscroll,* and *Leftscroll* move the entire screen display (with wraparound) one position in the appropriate direction for each call. Edit lines are included in the scrolling. The documentation is correct when it states "characters ... will wraparound ... except when *Border* is on."

When a border is present, it normally does not scroll; in reprinting itself, it erases any scrolled material it encounters. What the documentation fails to mention – the demo illustrates it admirably – is that there is a location to POKE that permits any border to scroll appropriately along with everything else. The required statement is: POKE 31743,0. *Now* watch it go!

Editprint changes the current print position to the first of the two edit lines. This change is for a single PRINT only; the following PRINT reverts to the normal placement unless *Editprint* is called again. *Square* is a handy utility that draws squares and rectangles according to two sets of coordinates – one specified by PLOT, the other by a pair of POKEs. The width of the line is one-half character.

Other miscellany regarding *Graphics*: *Graphics* routines use Slow mode only; and substantial error code recovery is implemented. These codes are nicely summarized, in tabular form, in the documentation. Error recovery is not available in *Toolkit*, where it is not nearly as important.

I discovered no program bugs, and – with minor exceptions – found the documentation quite satisfactory. When instructions for LOADing are precisely followed, all goes well. I particularly recommend *Graphics Kit* and its well-done demo program as helpful tools for exploring the Sinclair/Timex graphics capabilities.

Programmers Toolkit
Graphics Kit
Softsync, Inc.
P.O. Box 480, Murray Hill Station
New York, NY 10156
$14.95/each $25/both

©

THE WORLD INSIDE THE COMPUTER

The Computer Friend: Getting To Know You

Fred D'Ignazio, Associate Editor

In my recent **COMPUTE!** columns (August through November 1982), I introduced the "computer friend." The friend is a little animated face that appears on the computer picture screen. A bell rings, and the friend "wakes up" and talks to the child. It asks the child's name. Then it asks the child to play some games. The friend program automatically calls up games (like the story-telling game in the November column). Each time the child is finished playing a game, the friend pops back on the screen.

Lots of computer *programs* are friendly. But few computer operating systems are friendly. The computer friend is my effort to make computers friendlier, warmer, and more human-like for little kids.

Children are going to be spending many, many hours in front of these machines. Often the child's interaction with the machine will replace his interaction with other people. If the child's interaction with the machine is cold and impersonal, the child will be losing valuable opportunities to develop social skills. The child's character may eventually begin to mirror the machine's.

Already many people are complaining about the effect of computers and video games on older kids. Computer "hackers" are stereotypically pictured as being antisocial loners who speak BASIC better than they speak English.

So what happens when little kids spend the same number of hours on their computers? Their values are still being formed. Their reservoir of social and emotional experiences is still relatively limited. Computers are certain to have a big effect on them. And it may be negative as well as positive. It is likely that the kids will become computer literate and enhance their mathematical, logical, and creative skills. But they may not experience enough of the interaction with adults and peers which is vital to their development.

Let's face it. Computers are great as mind expanders and sharpeners. But when it comes to charm and personality, computers are cold fish.

That's where the computer friend comes in. The friend is kin to the first computer languages developed in the 1950s. It is a very crude attempt to make computers friendlier and easier to use.

Before the early "high-level" languages appeared, people interacted with the computer on its own terms – in machine language. They spent hours, days, weeks translating complicated problems into endless strings of ones and zeros.

People got fed up dealing with the computer on *its* terms. After all, the machine was supposed to be the servant of human beings, not the other way around. People sought ways to get the computer to do its own translation. They developed the early compilers and interpreters that took English-like commands and translated them into the computer's binary language, and vice versa.

As a result, the computer became more human-like. It became easier to use and a lot more useful. As a machine that could almost speak English, it became a valuable sidekick for people who needed to solve problems.

A computer friend is a lot like the early computer languages. I think computer friends will be the next step in the computer's evolution. Lots of people will find a computer friend much more attractive than the "friendly" computers they're stuck with today. A human-like machine that

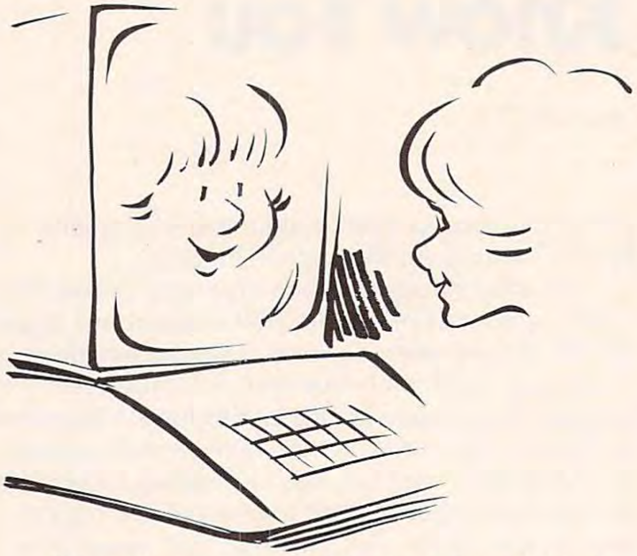
Fred D'Ignazio is a computer enthusiast and author of several books on computers for young people. He is presently working on two major projects: he is writing a series of books on how to create graphics-and-sound adventure games. He is also working on a computer mystery-and-adventure series for young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

focuses on a person and his or her problems will be a confidant, a colleague, and a comrade – not just a tool.

The Interested Friend

Here is the latest version of my computer friend program. This friend doesn't play games. It does only one thing: it learns about your child.



The older versions of the friend used Atari Graphics 2 Mode to enlarge the friend's face and the computer letters. However, that left less room on the screen for the friend's messages and questions, and the child's answers. In this version, we use the Graphics 0 Mode. This gives us plenty of space – up to 20 characters per line, and up to nine lines of text.

In the old program, on line 550, the friend jumped to a subroutine that asked if the child wanted to play games. In this program, the "game" GOSUB on line 550 has been replaced by a GOSUB to a subroutine that asks the child questions.

What sorts of questions? All sorts! For example:

Basic questions about the child's name, address, phone number, age, school, teachers, brothers, sisters, and pets.

Important questions like what the child likes to wear, what the child likes to eat, what makes the child happy or sad.

Whimsical questions like the child's favorite superhero, the scariest monster, and the names of any imaginary friends.

The only limitations are that the child cannot have more than three brothers, three sisters, three pets, and three imaginary playmates. If your child has more, then you need to modify the subroutines on lines 5780 through 5971, and add new DATA statements to those found on lines 13421 through 13456.

The computer takes all of the child's answers and stores them in a long string (C$). Percent symbols (%) are used as delimiters between the answers.

When the child is finished answering the friend's questions, the friend thanks him or her and tells the child how impressive he or she is. "You are a neat person!" the friend exclaims.

If you have old versions of the friend on your computer, look first at the lines preceded by asterisks. These are the lines I added or modified to create the new version of the friend program.

This version of the friend is geared to a disk system. At the end of the program (line 5974), the friend saves the child's answers (C$) on a disk file called "CHILD."

With only a couple of changes, you can make the program store the file on a tape cassette. First, you should erase the old line 5975, then add a new 5975 and a 5978:

```
5975 GOSUB 2010:REM * FRIEND TELLS AB
     OUT TAPE
5978 OPEN #1,8,0,"C:"
```

Next, add four DATA statements on lines 10050 through 10053:

```
10500 DATA 3
10051 DATA GET,THE,TAPE,READY.,-1
10052 DATA (PRESS,PLAY,AND,RECORD,BUT
      TONS.),-1
10053 DATA AFTER,THE,BEEP,PRESS,RETUR
      N,-1
```

Now the friend will prompt the child when it is ready to save the child's file. The friend will tell the child to press the right buttons on the program recorder and press the RETURN button on the computer.

Next Month

Next month I hope to print some of the interesting letters I have been receiving in response to my December 1982 column on "Sexism and Children's software."

In an upcoming column we'll teach the friend how to converse with the child using some of the information stored in the CHILD "data base." We'll see how we can automatically create files for several children.

I welcome your letters. And I will make every attempt to write back. From now on, you can write to me directly:

Fred D'Ignazio
2117 Carter Road, SW
Roanoke, VA 24015

```
100 REM *** DIMENSION VARIABLES
110 DIM M$(20):REM * MESSAGE
115 DIM C$(1500):REM * CHILD INFO
117 FOR I=0 TO 1450 STEP 50:C$(I+1,I+
    50)="{50 SPACES}":NEXT I
118 C$=""
```

NEW MULTI-USER SOFTWARE LETS THE WHOLE FAMILY SHARE IN THE JOY OF LEARNING.

Is the personal computer doing all it can to help our children learn?

To some degree, no, although it's not fair to blame it entirely on the computer. After all, computers are only as good as their software.

How can we improve this situation?

A solution already exists. But first, some background.

Where personal computers fail.

For years, studies have shown that children learn more efficiently in group situations. Peer groups, for example, motivate slower learners to persevere. Groups of older and younger children encourage divergent thinking. Even the simple "group" of a parent and child promotes faster acceptance of new ideas by combining education with trust and confidence.

But personal computers and their programs are designed to be personal. One computer, one child. It's hard for anyone else to be part of the learning experience, even you.

At least not until today.

A simple solution.

When two educational researchers, Dr. Matilda Butler and Dr. William Paisley, observed this problem they proposed an interesting, yet simple, solution. Instead of writing programs that shut out brothers, sisters, friends, and parents, why not give everyone the opportunity to share learning simultaneously. This one idea sparked an entire line of unique educational programs and gave birth to a new company, Edupro.

Software that shares.

With Edupro's Microgroup™ computer programs, up to eight players work at solving math, language, social studies, or science problems which are presented as contests, races, and puzzles. The players work together, either competitively or cooperatively, as they race against time, each other, or both.

The *Math-Race* program, for example, converts your computer into an electronic race track where children compete to answer math problems and advance toward the finish line. *Picture-Play* encourages everyone to create pictures together, teaching both spatial relationships and the value of cooperation. And *Team-Work* combines both cooperation and competition by pitting two teams (of up to four players) against each other in a race to solve word and number puzzles.

For the first time, your personal computer can bring all the benefits of group learning into your home. With a little assist from Edupro.

Designed for the simplest computers.

These unique programs run on the Atari 400 or 800, two of the world's most popular home computers. Remember, these aren't game cartridges, they're full *computer programs*, designed by educators. All are available on floppy disk or cassette, and each one requires the minimum amount of computer memory (16K for cassette, 24K for disk). That means the simplest Atari computer can let your children share the learning experience with up to seven additional friends. Joysticks required for *Word-Draw*, *Math-Hunt*, and *Picture-Play*; paddles required for *Word-Race*, *Math-Race*, and *Team-Work*.

Trust your own experience.

At the fall 1982 Computer-Using Educators Conference hundreds of educators witnessed hands-on demonstrations of our programs. Many of them said that this was a most effective way to judge their potential. But we want to offer you an even better opportunity. One those educators missed.

Edupro
Multi-user Educational Software
Edupro
Multi-user Educational Software
Users Instruction Manual
Scotch
diskette

We want you and your children to experience this new way to learn. So choose one or more programs on either disk or cassette. Try them yourself. Watch your children get more excited about learning. Enjoy the thrill of sharing the experience with them. We know of no other software that can turn a personal computer into a tool for sharing the joy of learning.

Fill out the order form and see the results for yourself.

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

I want to share the joy of learning with my children. Please send me the programs I've indicated below. I understand that each program is available on either disk or cassette (my choice) and comes with a complete set of instructions and catalog listing over 50 programs. Plus a coupon good for a 10% discount on my next order.

Quantity	Program Description	# of Disk	# of Cassette
	STORYBOOK FRIENDS: Ages 5–9		
______	WORD-DRAW: Storybook People and Places	______	______
______	MATH-HUNT: Number Relationships	______	______
	AMERICAN THEMES: Ages 8–13		
______	TEAM-WORK: Social Studies	______	______
______	MATH-HUNT: American Years: Multiplication and Division	______	______
	THE WORLD AROUND US: Ages 12–Adult		
______	WORD-DRAW: Science	______	______
______	MATH-RACE: Powers and Roots	______	______
	JUST FOR FUN: All Ages		
______	PICTURE-PLAY	______	______

Total #		Total Amount $
______	programs on disk @ $24.95 each	______
______	programs on cassette @ $19.95 each	______
______	Picture-Play, disk @ $19.95	______
______	Picture-Play, cassette @ $14.95	______
	CA residents add sales tax	______
	Total	______

My check or money order is enclosed for $______

Please bill ______MasterCard ______Visa

______ ______
(card no.) (exp. date)

Name ______

Address ______

City______ State ______ Zip ______

Signature ______

Allow 3 weeks for delivery. Satisfaction guaranteed.

Send to: Edupro, Dept. C01, P.O. Box 51346, Palo Alto, CA 94303.
Write to above address for brochure/ catalog listing
or phone inquiries: (415) 494-2790.

Edupro™

Edupro, P.O. Box 51346 Palo Alto, CA 94303, (415) 494-2790

```
120 N=1:REM * MESSAGE POINTER
125 DATNUM=10000:REM * WHERE TO READ
    DATA STATEMENTS
130 DIM NAME$(20):REM * CHILD'S NAME
500 REM *** FRIEND MASTER
510 IF PEEK(1791)=1 THEN GOSUB 7010:G
    OTO 550
515 GOSUB 1010:REM * FRIEND WAKE-UP
520 GOSUB 2010:REM * FRIEND INTRODUCE
    S HIMSELF/HERSELF
530 ANSWER=2:GOSUB 3210:REM * FRIEND
    LEARNS CHILD'S NAME
540 GOSUB 2010:REM * FRIEND HAPPY TO
    SEE CHILD
550 GOSUB 5610:REM * ASK INTRO QUESTI
    ONS
600 PRINT "{CLEAR}":POKE 752,0:END
1000 REM *** FRIEND WAKE-UP
1010 GOSUB 5010:REM * DRAW FACE
1020 GOSUB 5410:REM * DRAW SLEEP EYES
1030 GOSUB 5210:REM * DRAW CLOSED MOU
     TH
1035 FOR P=1 TO 800:NEXT P
1040 GOSUB 4010:REM * WAKE-UP BELL
1050 GOSUB 5460:REM * DRAW OPEN EYES
1060 FOR P=1 TO 600:NEXT P
1070 GOSUB 5320:REM * WINK EYE
1080 FOR P=1 TO 50:NEXT P
1085 GOSUB 5460:REM * DRAW OPEN EYES
1090 M=0:GOSUB 4820:REM * WINK NOISE
1100 FOR P=1 TO 800:NEXT P
1120 RETURN
2000 REM *** FRIEND TALK
2010 RESTORE DATNUM+N*10:REM * SELECT
      MESSAGE
2011 N=N+1:REM * SET POINTER TO NEXT
     SET OF FRIEND MESSAGES
2012 READ SNUM:REM * SNUM = NUMBER OF
      SCREENS IN CURRENT SET OF FRIEN
     D MESSAGES
2015 FOR K=1 TO SNUM
2020 GOSUB 3010:REM * FRIEND TALK--1
     SCREEN
2033 FOR P=1 TO 200:NEXT P
2035 GOSUB 5510:REM * CLEAR MESSAGE W
     INDOW
2040 NEXT K
2050 RETURN
3000 REM *** FRIEND TALKING--1 SCREEN
3010 PY=6:REM * MESSAGE VERTICAL (Y)
     START LOCATION
3030 PX=30:REM * HORIZONTAL (X) CENTE
     R OF MESSAGE ON SCREEN
3040 READ M$
3050 IF M$="-1" THEN FOR P=1 TO 200:N
     EXT P:GOSUB 5510:RETURN
3051 IF M$="*" THEN M$=NAME$
3055 GOSUB 5260:REM * OPEN MOUTH
3060 POSITION INT(PX-(LEN(M$)/2)+0.5)
     ,PY:REM * CENTER LINE
3070 PRINT #6;M$
3075 GOSUB 4810:REM * FRIEND SOUND
3080 FOR P=1 TO 10:NEXT P:REM * KEEP
     MOUTH OPEN
3090 GOSUB 5210:REM * CLOSE MOUTH
3095 FOR P=1 TO 50:NEXT P:REM * KEEP
     MOUTH CLOSED
3100 PY=PY+2
3110 GOTO 3040
3200 REM *** FRIEND ASKS CHILD A QUES
     TION
3210 OPEN #1,4,0,"K:"
3212 M$=""
3215 POSITION 20,6
3217 FOR I=1 TO 20
```

```
3220 GET #1,A
3222 IF A=126 AND I=1 THEN 3220
3225 IF A=126 THEN GOSUB 3310
3230 IF A=155 THEN 3265
3240 PRINT #6;CHR$(A);
3250 M$(LEN(M$)+1)=CHR$(A)
3260 NEXT I
3265 FOR P=1 TO 75:NEXT P
3267 GOSUB 5510:REM * CLEAR MESSAGE W
     INDOW
3270 CLOSE #1
3280 GOSUB 3410:REM * EVALUATE ANSWER
3290 RETURN
3310 POSITION I+18,6:PRINT #6;" ";
3312 POSITION I+18,6
3315 M$(LEN(M$))=""
3317 I=I-1
3320 GET #1,A
3330 IF A<>126 THEN 3390
3350 IF I<2 THEN 3320
3360 GOTO 3310
3390 RETURN
3400 REM *** EVALUATE ANSWER
3410 ON ANSWER GOSUB 3510,3610
3420 RETURN
3500 REM *** NO NEED TO STORE ANSWER
3510 RETURN
3600 REM *** ANSWER=CHILD'S NAME
3610 NAME$=M$
3620 RETURN
4000 REM *** WAKE-UP BELL
4010 BEL=105:TIM=7.5:GOSUB 4040
4020 BEL=132:TIM=8.5:GOSUB 4040
4030 SOUND 0,0,0,0:RETURN
4040 VLM=15:INC=0.79+TIM/50
4050 SOUND 0,BEL,10,VLM
4060 VLM=VLM*INC
4070 IF VLM>1 THEN 4050
4080 RETURN
4800 REM *** FRIEND VOICE
4810 M=INT(RND(1)*51)+15
4820 FOR A=M+25 TO M STEP -8
4830 SOUND 0,A,10,10
4840 FOR T=1 TO 10
4850 NEXT T
4860 NEXT A
4875 SOUND 0,0,0,0
4880 RETURN
5000 REM *** FRIEND'S FACE
5010 GRAPHICS 0
5015 POKE 752,1
5020 PRINT "{CLEAR}"
5040 POSITION 9,7:PRINT #6;"
     {3 SPACES}*"
5050 POSITION 9,8:PRINT #6;"  / \"
5060 POSITION 9,9:PRINT #6;" ====="
5070 POSITION 9,10:PRINT #6;"/
     {5 SPACES}\"
5090 POSITION 8,11:PRINT #6;"<:  ^  :
     >"
5100 POSITION 9,14:PRINT #6;"\_____/"
5110 RETURN
5200 REM *** CLOSE MOUTH
5210 POSITION 9,12:PRINT #6;":
     {5 SPACES}:"
5220 POSITION 9,13:PRINT #6;": --- :"
5230 RETURN
5250 REM *** OPEN MOUTH
5260 POSITION 9,12:PRINT #6;": ___ :"
5270 POSITION 9,13:PRINT #6;": \_/ :"
5280 RETURN
5300 REM *** LEFT EYE WINK
5320 POSITION 9,10:PRINT #6;": O - :"
5330 FOR P=1 TO 150:NEXT P
5340 RETURN
```

```
5400 REM *** EYES ASLEEP
5410 POSITION 9,10:PRINT #6;": - - :"
5440 RETURN
5450 REM *** EYES AWAKE
5460 POSITION 9,10:PRINT #6;": O O :"
5470 RETURN
5500 REM *** CLEAR MESSAGE WINDOW
5510 FOR Y=6 TO 22 STEP 2
5520 POSITION 20,Y
5530 PRINT #6;"{19 SPACES}"
5540 NEXT Y
5550 RETURN
5600 REM *** ASK INTRO QUESTIONS
5610 ANSWER=1:REM * EVALUATE ANSWER A
     S PART OF THIS SUBROUTINE
5615 GOSUB 2010:REM * CAN FRIEND ASK
     CHILD QUESTIONS?
5620 GOSUB 3210:REM * GET CHILD'S ANS
     WER
5630 IF M$(1,1)<>"N" THEN 5640
5632 RESTORE 12526:SNUM=1:GOSUB 2015:
     REM * FRIEND SAYS GOOD-BYE!
5635 GOTO 5745
5640 IF M$(1,1)<>"Y" THEN N=N-1:GOTO
     5610
5650 RESTORE 13010
5660 READ SNUM:REM * READ # OF QUESTI
     ONS
5670 FOR I=1 TO SNUM
5680 GOSUB 3010:REM * FRIEND ASKS ONE
      QUESTION
5685 FOR P=1 TO 200:NEXT P
5688 GOSUB 5510:REM * CLEAR MESSAGE W
     INDOW
5690 GOSUB 3210:REM * GET CHILD'S ANS
     WER
5700 GOSUB 5755:REM * ADD ANSWER TO S
     TRING
5710 NEXT I
5715 GOSUB 5785:REM * BROTHERS?
5720 GOSUB 5845:REM * SISTERS?
5725 GOSUB 5895:REM * PETS?
5730 GOSUB 5945:REM * IMAGINARY FRIEN
     DS?
5740 N=4:GOSUB 2010:REM * NICE GETTIN
     G TO KNOW YOU!
5741 GOSUB 5975:REM * SAVE ANSWER
5742 RESTORE 12526:SNUM=1:GOSUB 2015:
     REM * FRIEND SAYS GOOD-BYE!
5745 RETURN
5750 REM *** ADD CHILD'S ANSWER TO ST
     RING
5755 IF LEN(C$)<>0 THEN C$(LEN(C$)+1)
     ="%"
5760 C$(LEN(C$)+1)=M$
5770 RETURN
5780 REM *** BROTHERS?
5785 RESTORE 13420
5790 GOSUB 3010:REM * ASK QUESTION
5795 GOSUB 3210:REM * GET ANSWER
5800 GOSUB 5755:REM * ADD ANSWER TO S
     TRING
5805 IF M$="0" OR M$="NONE" THEN RETU
     RN
5810 FOR I=1 TO VAL(M$)
5815 GOSUB 3010:REM * ASK QUESTION
5820 GOSUB 3210:REM * GET ANSWER
5825 GOSUB 5755:REM * ADD ANSWER TO S
     TRING
5827 GOSUB 3010:REM * ASK QUESTION
5829 GOSUB 3210:REM * GET ANSWER
5830 GOSUB 5755:REM * ADD ANSWER TO S
     TRING
5835 NEXT I
5837 RETURN
```

```
5840 REM *** SISTERS?
5845 RESTORE 13430
5849 GOSUB 3010:REM * ASK QUESTION
5851 GOSUB 3210:REM * GET ANSWER
5853 GOSUB 5755:REM * ADD ANSWER TO S
     TRING
5855 IF M$="0" OR M$="NONE" THEN RETU
     RN
5857 FOR I=1 TO VAL(M$)
5859 GOSUB 3010:REM * ASK QUESTION
5862 GOSUB 3210:REM * GET ANSWER
5864 GOSUB 5755:REM * ADD ANSWER TO S
     TRING
5866 GOSUB 3010:REM * ASK QUESTION
5868 GOSUB 3210:REM * GET ANSWER
5870 GOSUB 5755:REM * ADD ANSWER TO S
     TRING
5872 NEXT I
5875 RETURN
5890 REM *** PETS?
5895 RESTORE 13440
5900 GOSUB 3010:REM * ASK QUESTION
5902 GOSUB 3210:REM * GET ANSWER
5904 GOSUB 5755:REM * ADD ANSWER TO S
     TRING
5906 IF M$="0" OR M$="NONE" THEN RETU
     RN
5908 FOR I=1 TO VAL(M$)
5910 GOSUB 3010:REM * ASK QUESTION
5912 GOSUB 3210:REM * GET ANSWER
5914 GOSUB 5755:REM * ADD ANSWER TO S
     TRING
5916 GOSUB 3010:REM * ASK QUESTION
5918 GOSUB 3210:REM * GET ANSWER
5920 GOSUB 5755:REM * ADD ANSWER TO S
     TRING
5922 NEXT I
5924 RETURN
5940 REM *** IMAGINARY FRIENDS?
5945 RESTORE 13450
5947 GOSUB 3010:REM * ASK QUESTION
5949 GOSUB 3210:REM * GET ANSWER
5951 GOSUB 5755:REM * ADD ANSWER TO S
     TRING
5953 IF M$="0" OR M$="NONE" THEN RETU
     RN
5955 FOR I=1 TO VAL(M$)
5957 GOSUB 3010:REM * ASK QUESTION
5959 GOSUB 3210:REM * GET ANSWER
5961 GOSUB 5755:REM * ADD ANSWER TO S
     TRING
5963 GOSUB 3010:REM * ASK QUESTION
5965 GOSUB 3210:REM * GET ANSWER
5967 GOSUB 5755:REM * ADD ANSWER TO S
     TRING
5969 NEXT I
5971 RETURN
5974 REM *** SAVE ANSWER
5975 OPEN #1,8,0,"D:CHILD"
5980 PRINT #1;C$
5990 CLOSE #1
5995 RETURN
6000 REM *** FRIEND'S GAMES
6010 GOSUB 2010:REM * FRIEND ASKS CHI
     LD: PLAY A GAME?
6020 ANSWER=1:GOSUB 3210:REM * GET CH
     ILD'S ANSWER
6030 IF M$(1,1)="N" THEN 6090
6040 IF M$(1,1)<>"Y" THEN N=N-1:GOTO
     6010
6050 GOSUB 6110:REM * SELECT GAME
6090 RESTORE 12526:SNUM=1:GOSUB 2015:
     REM * FRIEND SAYS GOOD-BYE!
6095 RETURN
6100 REM *** SELECT GAME
```

```
6110 DATNUM=12000:N1=N:N=1:REM * RESE
     T DATA POINTERS
6120 GOSUB 2010:REM * GENIE BEGINS GA
     ME-SELECTION QUESTION
6130 READ GAMENUM
6140 N=N+1
6150 FOR Z=1 TO GAMENUM
6160 GOSUB 2010:REM * DISPLAY GAME NA
     ME
6170 GOSUB 3210:REM * GET CHILD'S ANS
     WER
6180 IF M$(1,1)="Y" THEN GOSUB 6310:G
     OTO 6410
6190 IF M$(1,1)<>"N" THEN N=N-1:GOTO
     6160
6200 NEXT Z
6210 N=52:GOSUB 2010:REM * NO GAMES S
     ELECTED--FRIEND'S SORRY MESSAGE
6220 RETURN
6300 REM *** PREPARE FRIEND'S MEMORY
     FOR EXIT FROM FRIEND PROGRAM
6301 REM *** STORE CHILD'S NAME
6302 REM *** IN LOCATIONS
6303 REM *** 1781-1789
6304 REM *** (LENGTH OF NAME IN 1790)
6305 REM *** AND SET LOCATION 1791
6306 REM *** AS FLAG THAT
6307 REM *** FRIEND HAS ALREADY
6308 REM *** BEEN CALLED SINCE
6309 REM *** TURNING ON COMPUTER
6310 REM
6315 FOR I=1 TO LEN(NAME$)
6320 POKE 1780+I,ASC(NAME$(I,I))
6330 NEXT I
6335 POKE 1790,LEN(NAME$)
6340 POKE 1791,1
6350 RETURN
6400 REM *** DISK VERSION OF FRIEND
6405 REM ***
6408 REM *** SELECT GAME PROGRAM/EXIT
      FRIEND
6410 GOTO 6410+Z*10
6420 RUN "D:TELLTALE"
7000 REM *** FRIEND CALLED ON BEFORE
7010 FOR I=1 TO PEEK(1790)
7020 NAME$(LEN(NAME$)+1)=CHR$(PEEK(17
     80+I))
7030 NEXT I
7040 GOSUB 5010:GOSUB 5210:GOSUB 5460
     :REM * DRAW FRIEND
7050 DATNUM=11000:GOSUB 2010:REM * NE
     W FRIEND MESSAGES
7060 DATNUM=10000:N=3
7070 RETURN
10000 REM *** WAKE-UP FRIEND
10005 REM *** MESSAGES
10010 DATA 3
10011 DATA HI, I'M, GED,-1
10012 DATA YOU,TURNED,ME,ON,-1
10013 DATA WHO'S,OUT,THERE?,-1
10020 DATA 2
10021 DATA I'M,SO,HAPPY,-1
10022 DATA TO,SEE,YOU,*,-1
10030 DATA 2
10031 DATA CAN,I,ASK,YOU,-1
10032 DATA SOME,QUESTIONS?,-1
10040 DATA 3
10041 DATA THANKS,*,-1
10042 DATA I'M GLAD,I LEARNED,ABOUT,Y
      OU.,-1
10043 DATA YOU,ARE,A,NEAT,PERSON!,-1
11000 REM *** FRIEND ALREADY AWAKE ME
      SSAGES
11010 DATA 5
11011 DATA HI,*,-1
```

```
11012 DATA I,HOPE,YOU,-1
11013 DATA HAD,FUN!!,-1
11014 DATA I,WONDER,WHAT,-1
11015 DATA WE,SHOULD,DO,NOW.,-1
12000 REM *** GAMES
12001 REM
12002 REM *** LIST GAMES ON
12003 REM *** EVERY 10TH LINE--
12004 REM *** LINES 12030-12490
12005 REM *** FOR A MAXIMUM OF
12006 REM *** 50 GAMES.
12007 REM
12010 DATA 2
12011 DATA DO,YOU,WANT,-1
12012 DATA TO,PLAY,-1
12020 DATA 1
12030 DATA 1
12031 DATA THE,STORY,GAME?,-1
12520 DATA 6
12521 DATA *,I,AM,SORRY,-1
12522 DATA NONE,OF,THE,GAMES,-1
12523 DATA LOOKED,FUN.,-1
12524 DATA MAYBE,WE,CAN,-1
12525 DATA PLAY,LATER.,-1
12526 DATA BYE!,BYE!,BYE!,-1
13000 REM *** QUESTIONS
13010 DATA 40
13015 DATA WHAT,IS,YOUR,FIRST,NAME?,-1
13020 DATA WHAT,IS,YOUR,MIDDLE,NAME?,
      -1
13030 DATA WHAT,IS,YOUR,LAST,NAME?,-1
13040 DATA WHAT,IS,YOUR,NICKNAME?,-1
13050 DATA WHAT,IS,YOUR,STREET,NAME?,
      -1
13060 DATA WHAT,IS,YOUR,STREET,NUMBER
      ?,-1
13070 DATA WHAT,IS,YOUR,APARTMENT,NUM
      BER?,-1
13080 DATA WHAT,IS,YOUR,APARTMENT,NAM
      E?,-1
13090 DATA WHAT,IS,THE,NAME,OF,YOUR,C
      ITY?,-1
13100 DATA WHAT,IS,THE,NAME,OF,YOUR,S
      TATE?,-1
13110 DATA WHAT,IS,YOUR,ZIP,CODE?,-1
13120 DATA WHAT,IS,YOUR,PHONE,NUMBER?
      ,-1
13130 DATA WHEN,IS,YOUR,BIRTHDAY?,-1
13140 DATA WHAT,YEAR,WERE,YOU,BORN?,-1
13150 DATA HOW, OLD,ARE,YOU?,-1
13160 DATA WHAT,IS,YOUR,FAVORITE,COLO
      R?,-1
13170 DATA WHAT,IS,YOUR,FAVORITE,TV,S
      HOW?,-1
13180 DATA WHAT,IS,YOUR,FAVORITE,MOVI
      E?,-1
13190 DATA WHAT,IS,YOUR,FAVORITE,BOOK
      ?,-1
13200 DATA WHAT,IS,YOUR,FAVORITE,VIDE
      OGAME?,-1
13210 DATA WHAT,IS,YOUR,FAVORITE,SPOR
      T?,-1
13220 DATA WHAT,IS,YOUR,FAVORITE,THIN
      G,TO,DO?,-1
13230 DATA WHAT,IS,THE,YUCKIEST,THING
      ,TO,DO?,-1
13240 DATA WHAT,IS,YOUR,FAVORITE,THIN
      G,TO,WEAR?,-1
13250 DATA WHAT,IS,YOUR,FAVORITE,HOLI
      DAY?,-1
13260 DATA WHAT,IS,THE,THING,YOU,LIKE
      ,MOST,ABOUT,YOURSELF?,-1
13270 DATA WHAT,TRICK,OR,SKILL,ARE,YO
      U,MOST,PROUD,OF?,-1
13280 DATA WHAT,IS,THE,SCARIEST,MONST
```

```
      ER,YOU,KNOW,OF?,-1
13290 DATA WHAT,IS,THE,NAME,OF,YOUR,B
      EST,FRIEND?,-1
13300 DATA WHO,IS,YOUR,FAVORITE,HERO?
      ,-1
13310 DATA WHO,IS,YOUR,FAVORITE,SUPER
      HERO?,-1
13320 DATA WHAT,MAKES,YOU,THE,HAPPIES
      T?,-1
13330 DATA WHAT,MAKES,YOU,THE,SADDEST
      ?,-1
13340 DATA WHAT DO,YOU WANT,TO BE,WHE
      N YOU,GROW UP?,-1
13350 DATA WHAT,IS,THE,NAME,OF,YOUR,S
      CHOOL?,-1
13360 DATA WHAT,IS,YOUR,TEACHER'S,NAM
      E?,-1
13370 DATA WHAT,GRADE OR CLASS,ARE,YO
      U,IN?,-1
13380 DATA WHAT,IS,THE NAME,OF,YOUR,F
      AVORITE, DOLL OR TOY?,-1
13390 DATA WHAT TYPE,OF,WORK,DOES,YOU
      R MOM,DO?,-1
13400 DATA WHAT TYPE,OF,WORK,DOES,YOU
      R DAD,DO?,-1
13420 DATA HOW,MANY,BROTHERS,DO YOU,H
      AVE?,-1
13421 DATA WHAT IS,BROTHER #1'S,NAME?
      ,-1
13422 DATA WHAT IS,BROTHER #1'S,AGE?,
      -1
13423 DATA WHAT IS,BROTHER #2'S,NAME?
      ,-1
13424 DATA WHAT IS,BROTHER #2'S,AGE?,
      -1
13425 DATA WHAT IS,BROTHER #3'S,NAME?
      ,-1
13426 DATA WHAT IS,BROTHER #3'S,AGE?,
      -1
13430 DATA HOW,MANY,SISTERS,DO YOU,HA
      VE?,-1
13431 DATA WHAT IS,SISTER #1'S,NAME?,
      -1
13432 DATA WHAT IS,SISTER #1'S,AGE?,-1
13433 DATA WHAT IS,SISTER #2'S,NAME?,
      -1
13434 DATA WHAT IS,SISTER #2'S,AGE?,-1
13435 DATA WHAT IS,SISTER #3'S,NAME?,
      -1
13436 DATA WHAT IS,SISTER #3'S,AGE?,-1
13440 DATA HOW,MANY,PETS,DO YOU,HAVE?
      ,-1
13441 DATA WHAT,KIND OF,ANIMAL,IS,PET
       #1?,-1
13442 DATA WHAT IS,PET #1'S,NAME?,-1
13443 DATA WHAT,KIND OF,ANIMAL,IS,PET
       #2?,-1
13444 DATA WHAT IS,PET #2'S,NAME?,-1
13445 DATA WHAT,KIND OF,ANIMAL,IS,PET
       #3?,-1
13446 DATA WHAT IS,PET #3'S,NAME?,-1
13450 DATA HOW MANY,IMAGINARY FRIENDS
      ,DO YOU,HAVE?,-1
13451 DATA WHAT,KIND OF,CREATURE,IS,F
      RIEND #1?,-1
13452 DATA WHAT IS,FRIEND #1'S,NAME?,
      -1
13453 DATA WHAT,KIND OF,CREATURE,IS,F
      RIEND #2?,-1
13454 DATA WHAT IS,FRIEND #2'S,NAME?,
      -1
13455 DATA WHAT,KIND OF,CREATURE,IS,F
      RIEND #3?,-1
13456 DATA WHAT IS,FRIEND #3'S,NAME?,
      -1
```

©

LETTER AND NUMBER PLAY

Garold R. Stone

This has given my two year old, Jesse, and me a good deal of fun together. It really isn't a game. It offers an opportunity to play with large letters and numbers on the screen.

When the program starts, it's in the Alphabet mode. A large letter "A" appears in the middle of the screen, and a small reverse video "A" appears near the bottom. Each time Jesse presses the space bar, the next letter in the alphabet replaces the previous one in the middle of the screen, and the new letter is added to a growing alphabetic sequence at the bottom.

I had originally planned to use a speech synthesizer with the program. But, while play testing it with Jesse, I discovered that I made an even better speech synthesizer. Although Jesse already can say his ABCD's, he is just now learning that the ABC song he has learned is really made up of things called letters.

He sits on my lap and presses the space bar to see the letters. I say the names of the letters, and he repeats after me (sometimes). Or I may ask him questions like "What is the first letter of the alphabet?" or "Can you find the *A*?" It's all quite relaxed, but he is being exposed to the names, shapes, and alphabetical order of the letters.

If he says the wrong name for a letter, I don't say, "That's not *B*." I simply say the correct name of the letter. He sometimes just sits and listens as I say the names, while he presses the space bar to advance the letters.

One day he asked to see the *Q* when we were only up to *D*. Later, I added code which lets him put any letter at the top of the screen by pressing its key on the keyboard. After pressing individual letter keys, he can continue the alphabet at the bottom of the screen from where we left off by pressing the space bar again.

At any time I can press the shift and space bar to start over with the letter *A*. If we ever get to the end of the alphabet, the string of letters at the bottom of the screen flashes ten times, and I make a big deal of it.

But there are lots of other things to do, too. He can guess the name of the next letter or try to find a letter on the keyboard. Sometimes he just wants to see some favorite letters and touch them on the screen.

Finally, I added the Numbers Game. To play with the numbers, press the SHIFT and the number one. Pressing the space bar displays the next higher number in large print in the middle of the screen. Numbers greater than 9999 will not fit on the screen. Pressing any of the digits, zero to nine, displays that digit in large print at the top of the screen. To start counting over with one again, press SHIFT and space. To get back to the alphabet, press SHIFT and *A*.

These sessions are not very long – three to five minutes, at most. Jesse is usually a little impatient as the program loads from tape. Just about the time it's ready, he's decided he wants to play another favorite game (another excuse for getting a disk drive).

Jesse has been playing at my computer since he was about 18 months old. By now he knows that you "press" the keys, not pound them, though exuberance can lead to banging. He has learned that the keyboard has letters, numbers, and symbols like "star" (*) and "arrow" (↑).

(I can provide a copy of the PET Version of the program to those sending a cassette with $3 and a SASE mailer to: Garold R. Stone, P.O. Box 153, Annapolis Junction, MD 20701.)

Program 1: PET/CBM Version

Some characters are inaccessible from the PET/CBM business style keyboard and adjustments will need to be made in order to run Letter and Number Play on this machine.

```
100 POKE59468,12:REM GRAPHICS MODE
110 PRINT CHR$(142)
120 PRINT"{CLEAR}";
130 PRINT"{03 DOWN}FOR THE SUPERVISING ADU
    LT:"
140 PRINT:PRINT"PRESS [SHIFT] AND [A] FOR ~
    THE ALPHABET:"
150 PRINT:PRINT"  PRESS LETTER KEYS OR [SP
    ACE] TO PLAY."
160 PRINT"  [SHIFT] & [A] RESETS ALPHABET ~
    TO 'A'."
```

The Light Pen at the Right Price:

Shown actual size.

$29.95 Each

SPECIAL INTRODUCTORY OFFER!

Less is more. This maxim has never been more true than now with the introduction of our new Edumate Light Pen. This affordable and reliable tool was originally designed and developed for use with our Learning Center educational software—however, it is the perfect accessory for your Atari 400/800, VIC-20 or Commodore 64, regardless of application. Response has been so overwhelming that we now announce a new price schedule for quantity orders:

1-4—$29.95 each

5-24—$20.97 each 25-99—$19.48 each

100 and more—$17.97 each

Order now! See your local dealer or order direct.
New catalog $2.00. Visa and MasterCard accepted—
please add $2.00 for postage and handling.
Call toll free!

1-800-334-SOFT

DEALER INQUIRIES INVITED

programmer'sinstitute

a division of **FUTURE HOUSE** — dept. c
p.o. box 3470, chapel hill, north carolina 27514, 919-967-0861

```
170 PRINT:PRINT:PRINT"PRESS [SHIFT] AND [1
    ] FOR THE NUMBERS:"
180 PRINT:PRINT"  PRESS NUMBER KEYS OR [SP
    ACE] TO PLAY."
190 PRINT"  [SHIFT] & [1] RESETS NUMBERS T
    O '1'."
200 PRINT:PRINT:PRINT"PRESS [SPACE] TO CON
    TINUE, '/' TO STOP"
210 REM STORE LETTERS IN A$()
220 DIM A$(26):DIM N$(9)
230 NL$="{DOWN}{Ø8 LEFT}":REM 1 DOWN 8 BAC
    K
240 FOR I=1 TO 26
250 REM SET UP A LETTER
260 FOR J=1 TO 6
270 READ A$:A$(I) = A$(I) + A$
280 NEXT J
290 NEXT I
300 GOSUB222Ø:REM SET UP NUMBERS
310 GETA$:IFA$=""THEN31Ø
320 GOSUB 317Ø
330 REM LETTERS
340 L$ = "":L=1:GOSUB 2Ø4Ø:GOSUB2Ø7Ø:GOSUB
     212Ø
350 GETA$:IF A$ = "" THEN 35Ø
360 GOSUB 317Ø
370 IFA$=CHR$(177)ORA$=CHR$(33)THENGOSUB29
    2Ø:GOTO34Ø:REM NUMBERS
380 IFA$ = CHR$(193) THEN 34Ø
390 IFA$=" "THEN L=L+1:IFL>26THEN34Ø
400 IFA$=" "THEN IF L>26 THEN 34Ø
410 IFA$=" "THEN GOSUB 2Ø4Ø:GOSUB2Ø7Ø:GOSU
    B 212Ø:GOTO35Ø
420 A = ASC(A$):T = L:REM REMEMBER L
430 IF A >= 65 AND A <= 9Ø THEN L=A-64:PRI
    NT"{HOME}";:GOSUB 2Ø7Ø
440 L=T
450 GOTO35Ø
460 PRINT"{CLEAR}";:END
470 REM LETTERS
480 DATA" N####M "
490 DATA"'     ' "
500 DATA"'     ' "
510 DATA"'#####P "
520 DATA"'     ' "
530 DATA"'     ' "
540 DATA" O####M "
550 DATA" %    ' "
560 DATA" %    N "
570 DATA" O####M "
580 DATA" %    ' "
590 DATA" L$$$$N "
600 DATA" N###M  "
610 DATA" %      "
620 DATA" %      "
630 DATA" %      "
640 DATA" %      "
650 DATA" M$$$N  "
660 DATA" O##M   "
670 DATA" %   M  "
680 DATA" %   '  "
690 DATA" %   '  "
700 DATA" %   N  "
710 DATA" L$$N   "
720 DATA" O####  "
730 DATA" %      "
740 DATA" L$$    "
750 DATA" %      "
760 DATA" %      "
770 DATA" L$$$$  "
780 DATA" O####  "
790 DATA" %      "
800 DATA" L$$    "
810 DATA" %      "
```

```
820 DATA" %      "
830 DATA" %      "
840 DATA" N##M   "
850 DATA" %      "
860 DATA" %$$$   "
870 DATA" %  '   "
880 DATA" %  '   "
890 DATA" M$$N   "
900 DATA" %   '  "
910 DATA" %   '  "
920 DATA" L$$$:  "
930 DATA" %   '  "
940 DATA" %   '  "
950 DATA" %   '  "
960 DATA" #P##   "
970 DATA"  '     "
980 DATA"  '     "
990 DATA"  '     "
1000 DATA"  '     "
1010 DATA" $:$$   "
1020 DATA"  #P##  "
1030 DATA"   '    "
1040 DATA"   '    "
1050 DATA"   '    "
1060 DATA" % '    "
1070 DATA" M$N    "
1080 DATA" '  N   "
1090 DATA" ' N    "
1100 DATA" 'N     "
1110 DATA" 'M     "
1120 DATA" ' M    "
1130 DATA" '  M   "
1140 DATA" %      "
1150 DATA" %      "
1160 DATA" %      "
1170 DATA" %      "
1180 DATA" %      "
1190 DATA" L$$$$  "
1200 DATA"'M    N%"
1210 DATA"' M  N %"
1220 DATA"'  MN  %"
1230 DATA"'      %"
1240 DATA"'      %"
1250 DATA"'      %"
1260 DATA"'M     %"
1270 DATA"' M    %"
1280 DATA"'  M   %"
1290 DATA"'   M  %"
1300 DATA"'    M %"
1310 DATA"'     M%"
1320 DATA"  N###M "
1330 DATA" N     M"
1340 DATA" %     '"
1350 DATA" %     '"
1360 DATA" M     N"
1370 DATA"  M$$$N "
1380 DATA"'#####M "
1390 DATA"'     ' "
1400 DATA"'$$$$$N "
1410 DATA"'       "
1420 DATA"'       "
1430 DATA"'       "
1440 DATA"  N###M "
1450 DATA" N     M"
1460 DATA" %     '"
1470 DATA" %     '"
1480 DATA" M    MN"
1490 DATA"  M$$$NM"
1500 DATA"'#####M "
1510 DATA"'     ' "
1520 DATA"'$$$$$N "
1530 DATA"'   M   "
1540 DATA"'    M  "
1550 DATA"'     M "
```

...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

```
1560 DATA" N####M "
1570 DATA" %      "
1580 DATA" M$$$$  "
1590 DATA"      M "
1600 DATA"      ' "
1610 DATA" M$$$$N "
1620 DATA" ##P### "
1630 DATA"   '    "
1640 DATA"   '    "
1650 DATA"   '    "
1660 DATA"   '    "
1670 DATA"   '    "
1680 DATA"        "
1690 DATA"'      %"
1700 DATA"'      %"
1710 DATA"'      %"
1720 DATA"'      %"
1730 DATA" M$$$$N "
1740 DATA"        "
1750 DATA"        "
1760 DATA"        "
1770 DATA" M    N "
1780 DATA"  M  N  "
1790 DATA"   MN   "
1800 DATA"        "
1810 DATA"'      %"
1820 DATA"'      %"
1830 DATA"'  NM  %"
1840 DATA"' N  M %"
1850 DATA"'N    M%"
1860 DATA" M    N "
1870 DATA"  M  N  "
1880 DATA"   MN   "
1890 DATA"   NM   "
1900 DATA"  N  M  "
1910 DATA" N    M "
1920 DATA" M    N "
1930 DATA"  M  N  "
1940 DATA"   MN   "
1950 DATA"   '    "
1960 DATA"   '    "
1970 DATA"   '    "
1980 DATA"  $$$$  "
1990 DATA"     N  "
2000 DATA"    N   "
2010 DATA"   N    "
2020 DATA"  N     "
2030 DATA"  ####  "
2040 REM PRINT LETTER
2050 PRINT"{CLEAR}";
2060 FOR I = 1 TO 9:PRINT:NEXT:RETURN
2070 PRINT"{15 RIGHT}";
2080 FOR I=0 TO 5
2090 PRINT MID$(A$(L),I*8+1,8);NL$;
2100 NEXT:RETURN
2110 REM PRINT ALPHABET
2120 L$ = L$ + CHR$(L+64)
2130 B$="{HOME}{21 DOWN}          {REV}":PRINT
   B$;L$
2140 FOR I=1 TO 250:NEXT
2150 IF L=26 THEN GOSUB2180
2160 FOR I=1 TO 10:GETA$:NEXT:RETURN
2170 REM FLASH ALPHABET
2180 FOR I=1 TO 10
2190 C$="{HOME}{21 DOWN}          ":PRINTC$;L$
   :FOR J=1 TO 100:NEXT
2200 PRINT B$;L$:FOR J=1 TO 100:NEXT
2210 NEXTI:RETURN
2220 REM STORE DIGITS IN N$()
2230 BL$="{DOWN}{03 LEFT}"
2240 FOR I=0 TO 9
2250 REM SET UP A DIGIT
2260 FOR J=0 TO 5
2270 READ N$
2280 N$(I)=N$(I)+N$
2290 NEXT J
2300 NEXT I:RETURN
2310 REM DIGITS
2320 DATA"N#M"
2330 DATA"% '"
2340 DATA"% '"
2350 DATA"% '"
2360 DATA"% '"
2370 DATA"M$N"
2380 DATA" N%"
2390 DATA"  %"
2400 DATA"  %"
2410 DATA"  %"
2420 DATA"  %"
2430 DATA" $L"
2440 DATA"N#M"
2450 DATA"  '"
2460 DATA" $N"
2470 DATA"N  "
2480 DATA"%  "
2490 DATA"L$$"
2500 DATA"N#M"
2510 DATA"  '"
2520 DATA" $N"
2530 DATA"  M"
2540 DATA"  '"
2550 DATA"M$N"
2560 DATA"% %"
2570 DATA"% %"
2580 DATA"% %"
2590 DATA"##O"
2600 DATA"  %"
2610 DATA"  %"
2620 DATA"O##"
2630 DATA"L$ "
2640 DATA"  M"
2650 DATA"  '"
2660 DATA"  '"
2670 DATA"M$N"
2680 DATA"N#M"
2690 DATA"%  "
2700 DATA"L$ "
2710 DATA"% M"
2720 DATA"% '"
2730 DATA"M$N"
2740 DATA"O#P"
2750 DATA"  N"
2760 DATA" N "
2770 DATA" % "
2780 DATA" % "
2790 DATA" % "
2800 DATA"N#M"
2810 DATA"% '"
2820 DATA"M$N"
2830 DATA"N M"
2840 DATA"% '"
2850 DATA"M$N"
2860 DATA"N#M"
2870 DATA"% '"
2880 DATA"M$:"
2890 DATA"  '"
2900 DATA"  '"
2910 DATA"M$N"
2920 N=1:GOSUB3160
2930 GETA$:IFA$=""THEN2930
2940 GOSUB 3170
2950 IFA$=" "THEN N=N+1:GOSUB3160:GOTO2930
2960 T=N:N=ASC(A$)-48
2970 IFN>=0ANDN<10THENGOSUB3020
2980 N=T
2990 IFA$=CHR$(177)OR A$=CHR$(33)THEN N=1:G
   OSUB3160
3000 IFA$=CHR$(193)THEN RETURN
```

You Can COUNT On Abacus Software

VIC * COMMODORE 64 * PET

*SPRITE-AID sprite editor with joystick option	$14.95
*SYNTHY-64 music & Sound Synthesizer.Fantastic!	$29.95
*SCREEN-GRAPHICS-64 add graphics commands to BASIC incl. sprites	$24.95
*SKIER-64 exciting gameware	$19.95
Tiny Basic Compiler for Vic, CBM-64 or Pet	$19.95
*BUDGETEER Visual planner for Vic, CBM-64 or Pet	$19.95
*VIC QUICK CHART presentation chartmaker	$14.95
*VIC GREAT BALLOON RACE another exciting game	$14.95
VIC I-CHING oriental fortune teller(16Kexp)	$24.95
VIC SUPER EXPANDER SCREEN DUMP prints your graphics	$14.95
VIC JOYSTICK PAINTER	$14.95
VIC OR PET VIGIL games language with 9 games	$29.95
VIC OR PET PIPER the music machine!	$19.95
VIC HIRES / MULTICOLOR GRAPHICS UTILITIES (no extra memory)	$19.95
VIC GRAPHVICS super full-screen graphics (req. 3K or 8K mem.exp.)	$24.95
VIC TINY PILOT educational language	$17.95
*VIC CRIBBAGE (req. 16 K exp.)	$14.95
VIC MACHINE LANGUAGE GUIDE (\$6.95 foreign)	$ 5.95
PET MACHINE LANGAGE GUIDE (\$7.95 foreign)	$ 6.95
PET TINY PASCAL PLUS + (reg. 32 K)	$39.95
BASIC REFERENCE CARD (\$2.00 foreign)	$ 1.50

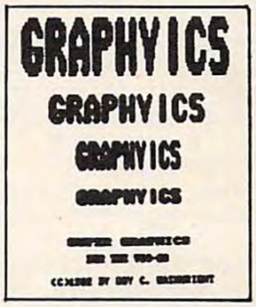
* NEW

Write for FREE Catalog or for fast service, call our Order Line.

Abacus Software

P.O. Box 7211, Grand Rapids, MI 49510 616 / 241-5510

All software packages come with complete instructions or manuals. Manuals ordered separately creditable towards purchase of software $5.00 each, $7.00 foreign. Add $3.00 per disk package. FREE POSTAGE in US and Canada. Foreign add $3.00 per item. Payment acceptable in US Dollars by check or international money order or via VISA, MC, ACCESS, Barclaycard.



TINY PILOT

Announcing . . . THE WHOLE PET CATALOG

A two year compendium of the **Midnite Software Gazette** and other resources for users of Commodore, CBM, PET, and VIC computers.

*The **Whole PET Catalog** contains:*

- **Over 500 independent reviews of commercial products.**
- **Over 700 education programs reviewed & organized by course.**
- **Over 200 reviews of free games.**
- **Information on over 1,800 free programs.**
- **Information about dozens of PET and VIC user groups.**
- **Many pages of hints and helps for all Commodore users.**
- **"Commodore's Family Tree", by Jim Butterfield.**
- **Completely reorganized and greatly expanded edition.**
- **Typeset and printed full-size on bond paper.**
- **In all, 320 pages of useful information.**

If you've seen **Midnite** before, directly or reprinted in the **TorPET** newsletter, here it is, complete in one volume, completely reorganized for easy reference, and greatly expanded with new information from members of the Toronto PET Users' Group.

"I still use my copy of **The Best of the PET Gazette** regularly. It was a treasure trove of information, and a great bargain for $10 three years ago. I hope you'll feel the same way about my **Whole PET Catalog** someday. Considering that it's three times as long, completely organized by topic, printed on bond paper instead of newsprint, typeset instead of dot matrix printed, bound instead of stapled, and still only $10, in spite of inflation, I'm sure you will." *—Jim Strasma, Contributing Editor, **Micro***

Whole PET Catalog $8

252 Bethlehem Pike
Colmar, PA 18915

215-822-7727

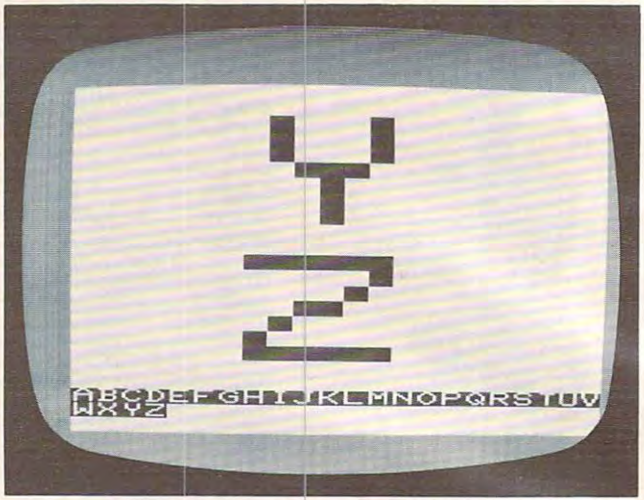
A B Computers

WRITE FOR CATALOG
Add $1.25 per order for shipping. We pay balance of UPS surface charges on all prepaid orders. Prices listed are on cash discount basis. Regular prices slightly higher. Prices subject to change.

```
3Ø1Ø GOTO293Ø
3Ø2Ø REM
3Ø3Ø P=1:F$=STR$(N)
3Ø4Ø IF LEN(F$)>5 THEN 46Ø
3Ø5Ø X=VAL(MID$(F$,P+1,1))
3Ø6Ø IFVAL(A$)=NTHENPRINT"{HOME}";:GOSUB311
     Ø:GOTO3Ø8Ø
3Ø7Ø GOSUB31ØØ
3Ø8Ø P=P+1:IFP<=LEN(F$)-1THEN3Ø5Ø
3Ø9Ø FORQ=1TO 25Ø:NEXT:FORQ=1TO9:GETA$:NEXT
     :RETURN
31ØØ PRINT"{HOME}{1Ø DOWN}";
311Ø PRINT"{2Ø RIGHT}";:FORQ=1TOLEN(F$):PRI
     NT"{Ø5 LEFT}";:NEXT
312Ø FORQ=1TO P:PRINT"{Ø5 RIGHT}";:NEXT
313Ø FOR I=Ø TO 5
314Ø PRINT MID$(N$(X),I*3+1,3);BL$;
315Ø NEXT:RETURN
316Ø PRINT"{CLEAR}";:GOSUB3Ø2Ø:RETURN
317Ø IF A$="/" THEN 46Ø
318Ø RETURN
```

Program 2: VIC Version

```
1ØØ PRINT"{CLEAR}{Ø9 DOWN}{RIGHT}{REV}LETT
    ERS AND NUMBERS"
11Ø PRINT"{Ø9 RIGHT}{REV}PLAY"
12Ø FOR I=1 TO2ØØØ
13Ø NEXT
14Ø PRINT"{CLEAR}";
15Ø PRINT:PRINT"PRESS [SHIFT] AND [A] FOR ~
    THE ALPHABET:"
16Ø PRINT:PRINT" *PRESS LETTER KEYS OR [SP
    ACE] TO PLAY."
17Ø PRINT"  [SHIFT] & [A] RESETS ALPHABET ~
    TO 'A'."
18Ø PRINT:PRINT:PRINT"PRESS [SHIFT] AND [1
    ] FOR THE NUMBERS:"
19Ø PRINT:PRINT" *PRESS NUMBER KEYS OR [SP
    ACE] TO PLAY."
2ØØ PRINT"  [SHIFT] & [1] RESETS NUMBERS T
    O '1'."
21Ø PRINT"{Ø2 DOWN}PRESS [SPACE] TO"
22Ø PRINT"CONTINUE, '/'TO STOP"
23Ø GETA$:IFA$=""THEN23Ø
24Ø GOSUB 89Ø
25Ø REM LETTERS
26Ø L$ = "":L=1:GOSUB 39Ø:GOSUB42Ø:GOSUB 5
    3Ø
27Ø GETA$:IF A$ = "" THEN 27Ø
28Ø GOSUB 89Ø
29Ø IFA$=CHR$(33)THENGOSUB64Ø:GOTO26Ø:REM ~
    NUMBERS
3ØØ IFA$ = CHR$(193) THEN 26Ø
31Ø IFA$=" "THEN L=L+1:IFL>26THEN26Ø
32Ø IFA$=" "THEN IF L>26 THEN 26Ø
33Ø IFA$=" "THEN GOSUB 39Ø:GOSUB42Ø:GOSUB ~
    53Ø:GOTO27Ø
34Ø A = ASC(A$):T = L:REM REMEMBER L
35Ø IF A >= 65 AND A <= 9Ø THEN L=A-64:PRI
    NT"{HOME}";:GOSUB 42Ø
36Ø L=T
37Ø GOTO27Ø
38Ø PRINT"{CLEAR}";:END
39Ø REM PRINT LETTER
4ØØ PRINT"{CLEAR}";
41Ø FOR I = 1TO 9:PRINT:NEXT:RETURN
42Ø T8=6
43Ø M=32768+8*L
44Ø PRINT"{Ø2 DOWN}";
45Ø FOR J=M TO M+7
46Ø D=PEEK(J):FOR K=1 TO 8
47Ø Y=146:D=D*2:IF D>255 THEN D=D-256:Y=18
```



A wrong letter match in "Letter And Number Play," VIC-20 version.

```
48Ø PRINT TAB(T8);CHR$(Y);CHR$(32);
49Ø NEXT K
5ØØ PRINTTAB(T8);CHR$(146)
51Ø NEXT J:RETURN
52Ø REM PRINT ALPHABET
53Ø L$ = L$ + CHR$(L+64)
54Ø B$="{HOME}{2Ø DOWN}{REV}":PRINT B$;L$
55Ø FOR I=1 TO 25Ø:NEXT
56Ø IF L=26 THEN GOSUB59Ø
57Ø FOR I=1 TO 1Ø:GETA$:NEXT:RETURN
58Ø REM FLASH ALPHABET
59Ø FOR I=1 TO 1Ø
6ØØ C$="{HOME}{2Ø DOWN}":PRINTC$;L$:FOR J=
    1 TO 1ØØ:NEXT
61Ø PRINT B$;L$:FOR J=1 TO 1ØØ:NEXT
62Ø NEXTI:RETURN
63Ø REM DIGITS
64Ø N=1:GOSUB88Ø
65Ø GETA$:IFA$=""THEN65Ø
66Ø GOSUB 89Ø
67Ø IFA$=" "THEN N=N+1:GOSUB88Ø: GOTO65Ø
68Ø T=N:N=ASC(A$)-48
69Ø IFN>=ØANDN<1ØTHENGOSUB74Ø
7ØØ N=T
71Ø IFA$=CHR$(33) THEN N=1:GOSUB88Ø
72Ø IFA$=CHR$(193)THEN RETURN
73Ø GOTO65Ø
74Ø REM
75Ø P=1:F$=STR$(N)
76Ø X=VAL(MID$(F$,P+1,1))
77Ø IFVAL(A$)=NTHENPRINT"{HOME}";:L=X:GOSU
    B82Ø:GOTO79Ø
78Ø L=X:GOSUB81Ø
79Ø P=P+1:IFP<=LEN(F$)-1THEN76Ø
8ØØ FORQ=1TO 25Ø:NEXT:FORQ=1TO9:GETA$:     ~
     NEXT:RETURN
81Ø PRINT"{HOME}{1Ø DOWN}";
82Ø T8=17:FORQ=1TOLEN(F$):T8=T8-7:NEXT
83Ø FORQ=1TOP:T8=T8+7:NEXT
84Ø IF T8<ØTHEN 38Ø
85Ø L=L+48
86Ø GOSUB 43Ø
87Ø RETURN
88Ø PRINT"{CLEAR}";:GOSUB74Ø:RETURN
89Ø IF A$="/" THEN 38Ø
9ØØ RETURN
```

Program 3: Apple Version

```
10  LOMEM: 16384
20  DIM L$(26)
```

AMERICAN PERIPHERALS

122 BANGOR ST.
LINDENHURST, N.Y. 11757
516-226-5849

V 6 FOUR ACROSS
Vertical game for two players; similar to tic-tac-toe. $9.95

V 10 SNAKE
Like the Arcade game SURROUND. One or two players. $9.95

V 12 BREAKOUT
Adaption of the classic Ping-Pong against the wall. All time favorite. $9.95

V 78 BRICK
The computer has a brick float across the screen and it disappears before it hits the window. You have to stop the brick before it breaks the glass. $7.95

V 79 SHOOT
You have to shoot a moving object. The game has five different skill levels. $7.95

V 80 REVERSE
This game is played with two sets of pegs. The board starts with a set of pegs at each end and you must reverse them. $7.95

V 81 CAPTURE
You have to capture the two beasts by containing them in the brick cage that you build. It has nine skill levels. $9.95

V 82 WORLD CAPITALS
An entertaining and educational variation of STATE CAPITALS. $9.95

V 83 JACKPOT
You must see this one armed bandit in action to believe it. Full color graphics and sound. $7.95

V 84 STATE CAPITALS
The computer displays a state or capital; the student types in the corresponding capital or state. Even corrects spelling mistakes. $9.95

V 85 HANGMAN
Unbelievable graphics and sound with a twist of humor. Kids of all ages will love this one. $9.95

V 86 TAC-TAC-TOE
The perennial favorite, a fast-paced colorful game. Try if you can to beat the VIC. $7.95

V 87 MEMORY
The VIC challenges your memory to the ultimate degree. Just like the old T.V. show Concentration. $9.95

V 88 MATCH
Hand and eye coordination are a must in this game. Beat the clock and get the maximum number of points. $7.95

V 89 MONKS
A devilish game of logic. It will have you glued to the VIC for days. $7.95

V 428 INTRUDER-SCRAMBLE
Just as fine as the arcade game. Machine language. A must! $19.95

V 90 MOSAICS
A variation of Rubics cube for the VIC. This game has never been seen before for any computer. We wrote the program but could not solve the puzzle. $9.95

V 91 SENKU
If you like thinking logically, this one will challenge you all the way! Based on the popular game of Mastermind. $7.95

V 92 MAGIC PAD
A mini-version of Visi-Calc for the VIC. Allows storage and retrieval of data from cassette; invaluable for personal and business use. Turns VIC into the perfect record keeper. $20.00

V107 BINARY NUMBERS
This program is designed to introduce you to the binary number system and the conversion to binary from base 10 numbers and back. $9.95

V108 BOMBER
You must decide who you want to fly for. You then get to pick a target and your experience level. $9.95

V151 BIZZ-BUZZ
Math game that tests the student on division by 2&3. Good for elementary school students. With color and sound. $9.95

V152 MISSILE COMMAND
You have three bases and you must destroy as many space ships as you can before you run out of missiles. $9.95

V153 TANK VS. UFO
The tank is moving back and forth along the base and you must shoot the UFO before it shoots you. $9.95

VIC=20

The friendly computer

V190 BIORHYTHM
Just like the biorhythm charts you find in books. $9.95

V191 BLACKJACK
Just like Blackjack in the casinos. $14.95

V192 BEST STRAIGHT LINE
This program finds the equation for the best straight line through the desired points on a graph. $9.95

V194 SNAKMAN
Pacman for the VIC. $24.95

V197 ASTROBASE-2001
Destroy the alien invaders from space as they attack your planet. Requires 3K memory expander. $9.95

V199 SUBROUTINES
The use of this standard programming technique allows you to save much room and effort. Typical uses are stressed. $14.95

V299 A CAT HAS NINE LIVES
You're an alley cat who is trying desparately to defend himself from unidentified deadly objects. Fast paced game. $7.95

V300 PSYCHIC MISSILES
The object of the game is to guess where the target will be, then fire the missile! This program will exercise your psychic ability. Requires
$9.95
By Richard Leiman

V301 BEECHA GOTCHA
Play "Beecha Gotcha." If the harpoon hits the monster fish, I "beecha." If the monster eats the boat, I "gotcha." Requires 3K expander. $9.95
By Richard Leiman

V302 AIR ATTACK
You must shoot down enemy aircrafts with your limited supply of missiles. $7.95

commodore

WRITE FOR FREE SOFTWARE CATALOG

Please send me the **A**
VIC 1982 CATALOG
AMERICAN PERIPHERALS
122 Bangor Street
Lindenhurst, NY 11757

NAME ______________________
ADDRESS ______________________
CITY ______________ STATE ______________
ZIP ______________ COMPANY ______________

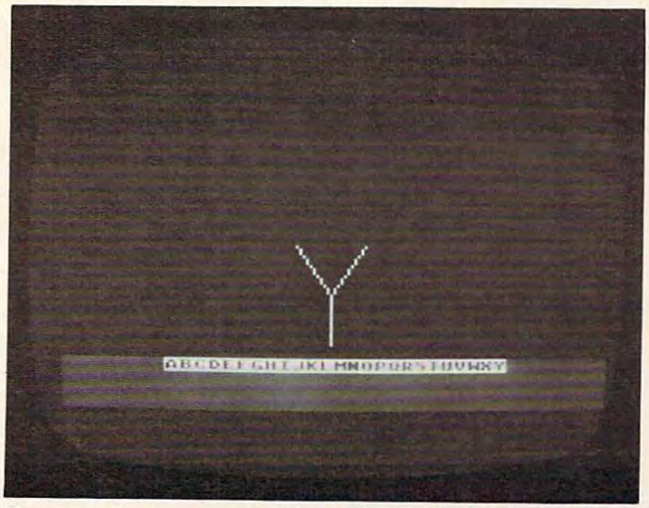
```
30  HOME
40  VTAB 3: PRINT "FOR THE SUPERVISING ADULT
   :"
50  PRINT : PRINT "PRESS <SHIFT> AND <A> FOR
     THE ALPHABET:"
60  PRINT : PRINT "  PRESS LETTER KEYS OR <S
     PACE> TO PLAY."
70  PRINT "  <SHIFT> & <A> RESETS ALPHABET T
     O 'A'."
80  PRINT : PRINT : PRINT "PRESS <SHIFT> AND
     <1> FOR THE NUMBERS:"
90  PRINT : PRINT "  PRESS NUMBER KEYS OR <S
     PACE> TO PLAY."
100  PRINT "  <SHIFT> & <1> RESETS NUMBERS T
     O '1'."
110  REM   STORE LETTER COORDINATES IN A
120  DIM A(26,20): DIM N(10,20)
130  REM  SET UP LETTERS
140  FOR I = 1 TO 26
150  FOR J = 1 TO 20
160  READ A(I,J)
170  NEXT J: NEXT I
180  GOSUB 1370: REM  SET UP NUMBERS
190  PRINT : PRINT : PRINT "PRESS <SPACE> TO
      CONTINUE, '/' TO STOP"
200  GET A$
210  GOSUB 2050
220  REM  LETTERS
230 L$ = "":L = 1: GOSUB 1170: GOSUB 1270
240  GET A$
250  GOSUB 2050
260  IF A$ =  CHR$ (33) THEN  GOSUB 1750:
     GOTO 230: REM  NUMBERS
270  IF A$ =  CHR$ (65) THEN 230
280  IF A$ = " " THEN L = L + 1:B = 0: IF L >
     26 THEN 230
290  IF A$ = " " THEN  IF L > 26 THEN 230
300  IF A$ = " " THEN  GOSUB 1170: GOSUB 127
     0: GOTO 240
310  IF B <  > 0 THEN T = L:L = B - 64: HCOL
    OR=0:Y7 = 30: GOSUB 1190:L = T: HCOLOR= 3
320 A =  ASC (A$):T = L: REM  REMEMBER L
330  IF A >  = 65 AND A <  = 90 THEN L = A -
    64:B = A:Y7 = 30: GOSUB 1190
340 L = T
350  GOTO 240
360  TEXT : HOME : END
370  REM  LETTERS
380  REM  ---A---
390  DATA  0,40,13,0,13,0,26,40,6,21
400  DATA  20,21,-1,-1,-1,-1,-1,-1,-1,-1
410  REM  ---B---
420  DATA  0,0,0,40,0,1,25,1,25,1
430  DATA  25,39,0,39,25,39,0,20,25,20
440  REM  ---C---
450  DATA  25,0,0,0,0,0,0,40,0,40
460  DATA  25,40,-1,-1,-1,-1,-1,-1,-1,-1
470  REM  ---D---
480  DATA  0,0,0,40,0,1,25,1,25,1
490  DATA  25,39,25,39,0,39,-1,-1,-1,-1
500  REM  ---E---
510  DATA  25,40,0,40,0,40,0,0,0,0
520  DATA  25,0,0,20,13,20,-1,-1,-1,-1
530  REM  ---F---
540  DATA  0,40,0,0,0,0,25,0,0,20
550  DATA  13,20,-1,-1,-1,-1,-1,-1,-1,-1
560  REM  ---G---
570  DATA  25,0,0,0,0,0,0,40,0,40
580  DATA  25,40,25,40,25,20,25,20,15,20
590  REM  ---H---
600  DATA  0,0,0,40,25,0,25,40,0,20
610  DATA  25,20,-1,-1,-1,-1,-1,-1,-1,-1
620  REM  ---I---
630  DATA  0,0,24,0,0,40,24,40,12,0
640  DATA  12,40,-1,-1,-1,-1,-1,-1,-1,-1
650  REM  ---J---
660  DATA  25,0,25,40,25,40,0,40,0,40
```

```
670  DATA  0,30,-1,-1,-1,-1,-1,-1,-1,-1
680  REM  ---K---
690  DATA  0,0,0,40,0,20,25,0,0,20
700  DATA  25,40,-1,-1,-1,-1,-1,-1,-1,-1
710  REM  ---L---
720  DATA  0,0,0,40,0,40,25,40,-1,-1
730  DATA  -1,-1,-1,-1,-1,-1,-1,-1,-1,-1
740  REM  ---M---
750  DATA  0,0,0,40,0,0,13,20,13,20
760  DATA  26,0,26,0,26,40,-1,-1,-1,-1
770  REM  ---N---
780  DATA  0,40,0,0,0,0,25,40,25,40
790  DATA  25,0,-1,-1,-1,-1,-1,-1,-1,-1
800  REM  ---O---
810  DATA  0,0,25,0,25,0,25,40,25,40
820  DATA  0,40,0,40,0,0,-1,-1,-1,-1
830  REM  ---P---
840  DATA  0,40,0,0,0,0,25,0,25,0
850  DATA  25,20,25,20,0,20,-1,-1,-1,-1
860  REM  ---Q---
870  DATA  0,0,25,0,25,0,25,40,25,40
880  DATA  0,40,0,40,0,0,20,35,30,45
890  REM  ---R---
900  DATA  0,40,0,0,0,0,25,0,25,0
910  DATA  25,20,25,20,0,20,10,20,25,40
920  REM  ---S---
930  DATA  25,0,0,0,0,0,0,20,0,20
940  DATA  25,20,25,20,25,40,25,40,0,40
950  REM  ---T---
960  DATA  0,0,25,0,13,0,13,40,-1,-1
970  DATA  -1,-1,-1,-1,-1,-1,-1,-1,-1,-1
980  REM  ---U---
990  DATA  0,0,0,40,0,40,25,40,25,40
1000  DATA  25,0,-1,-1,-1,-1,-1,-1,-1,-1
1010  REM  ---V---
1020  DATA  0,0,13,40,13,40,25,0,-1,-1
1030  DATA  -1,-1,-1,-1,-1,-1,-1,-1,-1,-1
1040  REM  ---W---
1050  DATA  0,0,5,40,5,40,13,0,13,0
1060  DATA  21,40,21,40,26,0,-1,-1,-1,-1
1070  REM  ---X---
1080  DATA  0,0,25,40,0,40,25,0,-1,-1
1090  DATA  -1,-1,-1,-1,-1,-1,-1,-1,-1,-1
1100  REM  ---Y---
1110  DATA  0,0,13,20,13,20,26,0,13,20
1120  DATA  13,40,-1,-1,-1,-1,-1,-1,-1,-1
1130  REM  ---Z---
1140  DATA  0,0,25,0,25,0,0,40,0,40
1150  DATA  25,40,-1,-1,-1,-1,-1,-1,-1,-1
1160  REM  PRINT LETTER
1170  HOME :Y7 = 90
1180  HGR : HCOLOR= 3
1190 X7 = 130
1200  FOR J = 1 TO 20 STEP 4
1210 X1 = A(L,J):Y1 = A(L,J + 1):X2 = A(L,J +
    2):Y2 = A(L,J + 3)
1220  IF X1 < 0 THEN 1240
1230  HPLOT X1 + X7,Y1 + Y7 TO X2 + X7,Y2 +
    Y7
1240  NEXT J
1250  RETURN
1260  REM  PRINT ALPHABET
1270 L$ = L$ +  CHR$ (L + 64)
1280  VTAB 22: HTAB 6: INVERSE : PRINT L$:
    NORMAL
1290  FOR I = 1 TO 250: NEXT
1300  IF L = 26 THEN  GOSUB 1330
1310  RETURN
1320  REM  FLASH ALPHABET
1330  VTAB 22: HTAB 6: FLASH : PRINT L$
1340  FOR I = 1 TO 3000: NEXT
1350  VTAB 22: HTAB 6: NORMAL : PRINT L$
1360  RETURN
1370  REM   STORE DIGIT COORDINATES IN N
1380  FOR I = 0 TO 9
1390  REM  SET UP A DIGIT
1400  FOR J = 0 TO 19
```

```
1410  READ N(I,J)
1420  NEXT J
1430  NEXT I: RETURN
1440  REM  DIGITS
1450  REM  ---0---
1460  DATA  0,0,20,0,20,0,20,40,20,40
1470  DATA  0,40,0,40,0,0,-1,-1,-1,-1
1480  REM  ---1---
1490  DATA  5,10,13,0,13,0,13,40,0,40
1500  DATA  26,40,-1,-1,-1,-1,-1,-1,-1,-1
1510  REM  ---2---
1520  DATA  0,10,12,0,12,0,24,10,24,10
1530  DATA  0,40,0,40,25,40,-1,-1,-1,-1
1540  REM  ---3---
1550  DATA  0,0,20,0,20,0,20,40,20,40
1560  DATA  0,40,0,20,20,20,-1,-1,-1,-1
1570  REM  ---4---
1580  DATA  20,0,0,35,0,35,25,35,20,0
1590  DATA  20,40,-1,-1,-1,-1,-1,-1,-1,-1
1600  REM  ---5---
1610  DATA  19,0,5,0,5,0,0,19,0,19
1620  DATA  20,19,20,19,20,40,20,40,0,40
1630  REM  ---6---
1640  DATA  2,0,0,20,0,20,22,20,22,20
1650  DATA  22,40,22,40,0,40,0,40,0,20
1660  REM  ---7---
1670  DATA  0,0,25,0,25,0,0,40,-1,-1
1680  DATA  -1,-1,-1,-1,-1,-1,-1,-1,-1,-1
1690  REM  ---8---
1700  DATA  0,0,0,40,0,40,20,40,20,40
1710  DATA  20,0,0,20,20,20,20,0,0,0
1720  REM  ---9---
1730  DATA  0,0,22,0,22,0,22,20,22,20
1740  DATA  0,20,0,20,0,0,22,20,20,40
1750 N1 = 1: GOSUB 2040
1760  GET A$: GOSUB 2050
1770  IF A$ = " " THEN N1 = N1 + 1:C2 = 0:
     GOSUB 2040: GOTO 1760
1780  IF C2 = 1 THEN X = N2: HCOLOR= 0:Y7 =
    30:X7 = 135: GOSUB 1980: HCOLOR= 3
1790 T = N1:N1 =  ASC (A$) - 48:N2 = N1:C2 =
    1: IF N1 < 0 OR N1 > 9 THEN N2 = 1
1800  IF N1 >  = 0 AND N1 < 10 THEN C5 = 1:
     GOSUB 1850
1810 N1 = T
1820  IF A$ =  CHR$ (33) THEN N1 = 1: GOSUB
    2040
1830  IF A$ =  CHR$ (65) THEN  RETURN
1840  GOTO 1760
1850  REM
1860 P = 1:F$ =  STR$ (N1)
1870 X =  VAL ( MID$ (F$,P,1))
1880  IF  VAL (A$) = N1 THEN Y7 = 30: GOSUB
    1930: GOTO 1900
1890  GOSUB 1920
1900 P = P + 1: IF P <  =  LEN (F$) THEN 187
    0
1910  RETURN
1920 Y7 = 90
1930 X7 = 135: FOR Q = 1 TO  LEN (F$):X7 = X
    7 - 33: NEXT
1940  FOR Q = 1 TO P:X7 = X7 + 33: NEXT
1950  IF P > 1 OR C5 = 1 THEN 1980
1970  HGR : HCOLOR= 3
1980  FOR J = 0 TO 19 STEP 4
1990 X1 = N(X,J):Y1 = N(X,J + 1):X2 = N(X,J +
    2):Y2 = N(X,J + 3)
2000  IF X1 < 0 THEN 2030
2010  HPLOT X1 + X7,Y1 + Y7 TO X2 + X7,Y2 +
    Y7
2020  NEXT
2030 C5 = 0: RETURN
2040  HOME : GOSUB 1850: RETURN
2050  IF A$ = "/" THEN 360
2060  RETURN
```

Program 4: Atari Version

```
50 OPEN #1,4,0,"K:"
60 DIM A$(1),L$(26),F$(4)
180 GRAPHICS 0:POKE 752,1
190 PRINT :PRINT :PRINT "FOR THE SUPE
    RVISING ADULT:"
200 PRINT :PRINT "PRESS [SHIFT] & [A]
     FOR THE ALPHABET:"
210 PRINT :PRINT "*PRESS LETTER KEYS
    OR [SPACE] TO PLAY."
211 PRINT " >[SHIFT]&[A] RESETS ALPHA
    BET TO 'A'."
220 PRINT :PRINT :PRINT "PRESS [SHIFT
    ] & [1] FOR THE NUMBERS:"
221 PRINT :PRINT "*PRESS NUMBER KEYS
    OR [SPACE] TO PLAY."
222 PRINT " >[SHIFT]&[1] RESETS NUMBE
    RS TO '1'."
250 PRINT :PRINT "{3 SPACES}WAIT FOR
    ARRAYS TO BE READ"
280 REM STORE LETTER COORDINATES IN A
290 DIM A(26,20):DIM N(10,20)
305 REM SET UP LETTERS
310 FOR L=1 TO 26
320 FOR L1=1 TO 20
330 READ A:A(L,L1)=A
340 NEXT L1
350 NEXT L
370 GOSUB 2270:REM SET UP NUMBERS
375 PRINT :PRINT :PRINT "PRESS [SPACE
    ] TO CONTINUE, '/' TO STOP"
380 GET #1,A
385 A$=CHR$(A):GOSUB 3500
390 REM LETTERS
400 L$="":L=1:GOSUB 2100:GOSUB 2170
410 GET #1,A:A$=CHR$(A):GOSUB 3500
420 IF A$=CHR$(33) THEN GOSUB 2970:GO
    TO 400:REM NUMBERS
430 IF A$=CHR$(65) THEN 400
440 IF A$=" " THEN L=L+1:B=0:IF L>26
    THEN 400
450 IF A$=" " THEN IF L>26 THEN 400
460 IF A$=" " THEN GOSUB 2100:GOSUB 2
    170:GOTO 410
465 IF B<>0 THEN T=L:L=B-64:COLOR 0:Y
    7=3:GOSUB 2120:L=T:COLOR 1
470 A=ASC(A$):T=L:REM REMEMBER L
480 IF A>=65 AND A<=90 THEN L=A-64:B=
    A:Y7=3:GOSUB 2120
490 L=T
500 GOTO 410
510 GRAPHICS 0:END
520 REM LETTERS
```



The computer awaits a match in "Letter And Number Play," Atari version.

```
530 REM ---A---
540 DATA 0,30,10,0,10,0,20,30,5,15
550 DATA 15,15,-1,-1,-1,-1,-1,-1,-1,-1
560 REM ---B---
570 DATA 0,0,0,30,0,1,20,1,20,1
580 DATA 20,29,0,29,20,29,0,15,20,15
590 REM ---C---
600 DATA 17,0,0,0,0,0,0,30,0,30
610 DATA 17,30,-1,-1,-1,-1,-1,-1,-1,-1
620 REM ---D---
630 DATA 0,0,0,30,0,1,17,1,17,1
640 DATA 17,29,17,29,0,29,-1,-1,-1,-1
650 REM ---E---
660 DATA 18,30,0,30,0,30,0,0,0,0
670 DATA 17,0,0,15,12,15,-1,-1,-1,-1
680 REM ---F---
690 DATA 0,30,0,0,0,0,17,0,0,15
700 DATA 12,15,-1,-1,-1,-1,-1,-1,-1,-1
710 REM ---G---
720 DATA 17,0,0,0,0,0,0,30,0,30
730 DATA 17,30,17,30,17,17,17,17,11,1
    7
740 REM ---H---
750 DATA 0,0,0,30,20,0,20,30,0,15
760 DATA 20,15,-1,-1,-1,-1,-1,-1,-1,-1
770 REM ---I---
780 DATA 0,0,20,0,0,30,20,30,10,0
790 DATA 10,30,-1,-1,-1,-1,-1,-1,-1,-1
800 REM ---J---
810 DATA 20,0,20,30,20,30,0,30,0,30
820 DATA 0,23,-1,-1,-1,-1,-1,-1,-1,-1
830 REM ---K---
840 DATA 0,0,0,30,0,15,15,0,0,15
850 DATA 15,30,-1,-1,-1,-1,-1,-1,-1,-
    1
860 REM ---L---
870 DATA 0,0,0,30,0,30,17,30,-1,-1
880 DATA -1,-1,-1,-1,-1,-1,-1,-1,-1,-
    1
890 REM ---M---
900 DATA 0,30,0,0,0,0,10,10,10,10
910 DATA 20,0,20,0,20,30,-1,-1,-1,-1
920 REM ---N---
930 DATA 0,30,0,0,0,0,20,30,20,30
940 DATA 20,0,-1,-1,-1,-1,-1,-1,-1,-1
950 REM ---O---
960 DATA 0,0,19,0,19,0,19,30,19,30
970 DATA 0,30,0,30,0,0,-1,-1,-1,-1
980 REM ---P---
990 DATA 0,30,0,0,0,1,16,1,16,1
1000 DATA 16,15,16,15,0,15,-1,-1,-1,-1
1010 REM ---Q---
1020 DATA 0,0,19,0,19,0,19,30,19,30
1030 DATA 0,30,0,30,0,0,15,25,23,35
1040 REM ---R---
1050 DATA 0,30,0,0,0,1,16,1,16,1
1060 DATA 16,15,16,15,0,15,5,15,16,30
1070 REM ---S---
1080 DATA 16,0,0,0,0,0,0,15,0,15
1090 DATA 16,15,16,15,16,30,16,30,0,3
     0
1100 REM ---T---
1110 DATA 0,0,20,0,10,0,10,30,-1,-1
1120 DATA -1,-1,-1,-1,-1,-1,-1,-1,-1,
     -1
1130 REM ---U---
1140 DATA 0,0,0,30,0,30,20,30,20,30
1150 DATA 20,0,-1,-1,-1,-1,-1,-1,-1,-
     1
1160 REM ---V---
1170 DATA 0,0,10,30,10,30,20,0,-1,-1
1180 DATA -1,-1,-1,-1,-1,-1,-1,-1,-1,
     -1
1190 REM ---W---
1200 DATA 0,0,5,30,5,30,10,0,10,0
1210 DATA 15,30,15,30,20,0,-1,-1,-1,-1
1220 REM ---X---
```

```
1230 DATA 0,0,20,30,0,30,20,0,-1,-1
1240 DATA -1,-1,-1,-1,-1,-1,-1,-1,-1,
     -1
1250 REM ---Y---
1260 DATA 0,0,10,15,10,15,20,0,10,15
1270 DATA 10,30,-1,-1,-1,-1,-1,-1,-1,
     -1
1280 REM ---Z---
1290 DATA 0,0,20,0,20,0,0,30,0,30
1300 DATA 20,30,-1,-1,-1,-1,-1,-1,-1,
     -1
2090 REM PRINT LETTER
2100 GRAPHICS 0:POKE 752,1:Y7=45
2110 GRAPHICS 6:POKE 752,1:COLOR 1
2120 X7=65
2130 FOR J=1 TO 20 STEP 4
2135 X1=A(L,J):Y1=A(L,J+1):X2=A(L,J+2
     ):Y2=A(L,J+3)
2140 IF X1<0 THEN 2147
2145 PLOT X1+X7,Y1+Y7:DRAWTO X2+X7,Y2
     +Y7
2147 NEXT J
2150 RETURN
2160 REM PRINT ALPHABET
2170 L$(LEN(L$)+1)=CHR$(L+192)
2180 PRINT "{5 SPACES}";L$
2200 IF L=26 THEN GOSUB 2230
2210 RETURN
2220 REM FLASH ALPHABET
2230 FOR I=1 TO 20:POKE 755,0:FOR W=1
      TO 20:NEXT W:POKE 755,2:FOR W=1
      TO 20:NEXT W:NEXT I
2250 RETURN
2270 REM STORE DIGITS IN N
2290 FOR I=0 TO 9
2300 REM SET UP A DIGIT
2310 FOR J=0 TO 19
2320 READ N:N(I,J)=N
2330 NEXT J:NEXT I
2350 RETURN
2360 REM DIGITS
2370 REM ---0---
2380 DATA 0,0,16,0,16,0,16,30,16,30
2390 DATA 0,30,0,30,0,0,-1,-1,-1,-1
2400 REM ---1---
2410 DATA 4,6,10,0,10,0,10,30,0,30
2420 DATA 20,30,-1,-1,-1,-1,-1,-1,-1,
     -1
2430 REM ---2---
2440 DATA 0,7,10,0,10,0,20,7,20,7
2450 DATA 0,30,0,30,20,30,-1,-1,-1,-1
2460 REM ---3---
2470 DATA 0,0,16,0,16,0,16,30,16,30
2480 DATA 0,30,0,15,16,15,-1,-1,-1,-1
2490 REM ---4---
2500 DATA 18,30,18,0,18,0,0,27,0,27
2510 DATA 20,27,-1,-1,-1,-1,-1,-1,-1,
     -1
2520 REM ---5---
2530 DATA 16,0,3,0,3,0,0,15,0,15
2540 DATA 17,15,17,15,17,30,17,30,0,3
     0
2550 REM ---6---
2560 DATA 5,0,0,15,0,15,16,15,16,15
2570 DATA 16,30,16,30,0,30,0,30,0,15
2580 REM ---7---
2590 DATA 0,0,20,0,20,0,0,30,-1,-1
2600 DATA -1,-1,-1,-1,-1,-1,-1,-1,-1,
     -1
2610 REM ---8---
2620 DATA 0,0,0,30,0,30,16,30,16,30
2630 DATA 16,0,16,0,0,0,0,15,16,15
2640 REM ---9---
2650 DATA 0,0,16,0,16,0,16,15,16,15
2660 DATA 0,15,0,15,0,0,16,15,10,30
2970 N1=1:GOSUB 3190
2980 GET #1,A:A$=CHR$(A):GOSUB 3500
```

LOGO

MIT MIT

MIT **$89.95** MIT

The Best and Most Complete LOGO for Apple II on the market at a Super Saver price.

Sprites now available for LOGO

Krell's College Board

SAT*

Preparation Series

ATARI, APPLE, PET, TRS-80, IBM

A COMPREHENSIVE PREPARATION PACKAGE
MORE THAN 40 PROGRAMS/$299.95

1. Diagnostic analysis
2. Prescription of Individual study plans
3. Coverage of all SAT* skills
4. Unlimited drill and practice
5. SAT* Exam Question simulator
6. All questions in SAT* format and at SAT* difficulty level
7. Instantaneous answers, explanations and scoring for problems
8. Worksheet generation and performance monitoring - (optional)
9. A complete record mmanagement system - (optional)
10. Systematic Instruction in pertinent math, verbal and test taking skills - (optional)

Krell's unique logical design provides personalized instruction for each student according to individual needs.

Shelby Lyman Chess

Shelby's Socrates Chess Tutorial Series uses the latest AI Techniques to customize each lesson for you. Construction modules cover every aspect of the game. For all micro's. Call or write for details.

Amazing Ben

The Royal Road to Artifical Intelligence

$79.95

CONNECTIONS

A complete game system. Learn the principles of scientific reasoning in your choice of game formats. Expand minds at all ages. Subject areas include: Geography, Biology, Everyday Objects, etc.

Game System $99.95

Data Base, $24.95 per subject, 3 for $50

Isaac Newton + F.G. Newton $49.95

Descarte's Delight $89.95

KRELL®
SOFTWARE CORP.

The state of the art in educational computing

1320 Stony Brook Road/Stony Brook, NY 11790
Telephone 516-751-5139

Krell Software Corp. has no official ties with the College Entrance Examination Board or the Educational Testing Service. Krell is, however, a supplier of products to the ETS

***Trademarks of Apple Comp. Corp., Tandy Corp. Commodore Corp. Digital Research Corp., I.B.M., Atari Corp.**

N.Y.S. residents add sales tax

Prices slightly higher outside U S

MasterCard VISA

TRACK REAL SATELLITES

LEARN CHINESE

MORTGAGE ANALYSIS

BUILD YOUR IQ

EXPLORE A JUNGLE

FLY TO THE MOON

CALENDAR

IMPROVE YOUR MEMORY

DRAW MURALS

PILOT A ROCKET

WHAT CAN I DO WITH MY COMPUTER?

PLAY CHESS WITH AN EXPERT

STOCK MARKET ANALYSIS

PLAY MUSIC

ASTROLOGY

BE AN AIR TRAFFIC CONTROLLER

BUDGET RECORD KEEPING

PROGRAM ROBOTS

RUN 3 MILE ISLAND

GO THRU PSYCHO-THERAPY

FLY AN AIRPLANE

CHRISTMAS CARD LISTS

WRITE LETTERS WITH SPELLING CORRECTIONS

DRIVE IN THE GRAND PRIX

LEARN TO TYPE

GENEALOGY

BEAT THE ODDS AT VEGAS

ALGEBRA

If you have been looking for new ways to use your Home Computer, a way to make your computer more versatile and exciting, then look no further!
THE SOFTWARE CONNECTION'S — PROGRAM OF THE MONTH CLUB is for you.

THE PROBLEM

In the past decade a revolution has occurred that has changed our lives -- the Computer Revolution. Every time we go to the store, watch television or even answer the telephone, computers are involved. What makes a computer perform all of these marvelous feats? The key is the PROGRAM

The Home Computer however, is still in its infancy. Sure you can buy a computer just about anywhere, but **the selection of programs offered at these places to make your computer perform marvelous feats for you is very limited.**

THE SOLUTION

There are creative and imaginative programmers worldwide writing programs for your computer. **With these programs your computer can transport you to a world never before available.** You can Fly to the Moon, Track Real Satellites, Improve Your Memory, Play Chess with an Expert or even Fly your own Airplane. Your computer can be used as a Paint Brush to Draw Murals or as a Musical Instrument to Compose and Play Music. With Financial Programs you can analyze the Stock and Bond Markets, Perform Mortgage Analysis, Balance Your Checkbook, or Control and Analyse your Household Budget. Using Educational Programs you can learn just about anything through a revolutionary technique called CAI (Computer-Assisted Instruction) or through Computer Simulations.

The key to all this is the PROGRAM.

The SOFTWARE CONNECTION's staff is seeking out the authors of these marvelous programs and are offering them to you through the

- PROGRAM OF THE MONTH CLUB -

Each month you will receive a description of a featured program along with 3 alternate programs - one from each catagory: **GAMES, EDUCATION and PERSONAL & FINANCE.** Take your time looking through it and decide if you want to add the featured program to your library. If not, check NO or one of the alternate programs on the enclosed card and return within 10 days. Otherwise be prepared to embark on a new Computer Adventure.

HOW TO GET STARTED

To get started complete the Club Membership Application telling us what kind of computer you have. Include the New Member Initiation Fee of $5.00 (½ OFF the Normal Initiation Fee).

As a **Special Offer** to readers of COMPUTE Magazine we will send you a certificate good for **10% OFF** all purchases made during any one month.

There are **No Minimum or Maximum** number of programs to purchase. All programs are fully tested and guaranteed to be a unique experience.

THE
SOFTWARE CONNECTION
— PROGRAM OF THE MONTH CLUB —

THE
SOFTWARE CONNECTION
— PROGRAM OF THE MONTH CLUB —
P.O. BOX 141319
DALLAS, TEXAS
75214

MEMBERSHIP APPLICATION

Yes. I want to join the Program of the Month Club. I understand that I will be sent a description of the featured program of the month along with three alternate programs, one from each category of Games, Education and Personal & Finance (Prices start at $14.95). Further, I understand that if I do not want the featured program I may select an alternate program or none at all by marking my desires on the enclosed card and returning it within 10 days. Otherwise, I will be ready to embark on a new and unique experience using my computer. I have enclosed the initiation fee of $5.00 (½ OFF the Normal Initiation Fee). I understand that there are no minimum or maximum number of purchases that I may make and that all programs are fully tested.

Name ____________________

Address ____________________ Apt No ______

City ____________________ State ______

Zip Code ____________________

Profession ____________________

□ Yes, I saw this ad in COMPUTE Magazine. Please send to me my certificate for 10% OFF all purchases made during any one month.

Ⓖ

COMPUTER CONFIGURATION

1) Computer Model* (Check 1)
- □ TRS 80 MODEL I - Level II
- □ TRS 80 MODEL III - Level II
- □ TRS 80 Color Computer
- □ Apple II with Integer Basic Firmware Card
- □ Apple II with Applesoft Firmware Card
- □ Apple II Language System
- □ Atari 400
- □ Atari 800
- □ VIC-20

2) Memory (Check 1)
- □ 16 K Bytes
- □ 24 K Bytes
- □ 32 K Bytes
  (TRS Disk Systems require a minimum of 32 K)
- □ 48 K Bytes
  (Apple Disk Systems require a minimum of 48 K)

3) Number of Disk Drives (Check 1)
- □ None
- □ 1
- □ 2 or More

4) Cassette Interface (Check 1)
- □ Yes
- □ No

5) Other Accessories (Check Maximum of 3)
- □ Color Monitor
- □ Printer
- □ Joysticks or Paddles

*Apple, Atari, VIC and TRS 80 are the trademarks of Apple Computer, Inc. Atari Inc., Commodore Int. and Tandy Corp. respectively

```
2990 IF A$=" " THEN N1=N1+1:C2=0:GOSU
     B 3190:GOTO 2980
2995 IF C2=1 THEN X=N2:COLOR 0:Y7=3:X
     7=70:GOSUB 3155:COLOR 1
3000 T=N1:N1=ASC(A$)-48:N2=N1:C2=1:IF
      N1<0 OR N1>9 THEN N2=1
3010 IF N1>=0 AND N1<10 THEN C5=1:GOS
     UB 3060
3020 N1=T
3030 IF A$=CHR$(33) THEN N1=1:GOSUB 3
     190
3040 IF A$=CHR$(65) THEN RETURN
3050 GOTO 2980
3060 REM LETTER PRINTING
3070 P=1:F$=STR$(N1)
3080 X=VAL(F$(P,P))
3090 IF ASC(A$)-48=N1 THEN Y7=3:GOSUB
      3140:GOTO 3110
3100 GOSUB 3130
3110 P=P+1:IF P<=LEN(F$) THEN 3080
3120 RETURN
3130 Y7=45
3140 X7=70:FOR Q=1 TO LEN(F$):X7=X7-2
     3:NEXT Q
3145 FOR Q=1 TO P:X7=X7+23:NEXT Q
3146 IF P>1 OR C5=1 THEN 3155
3150 GRAPHICS 6:COLOR 1
3155 FOR J=0 TO 19 STEP 4
3160 X1=N(X,J):Y1=N(X,J+1):X2=N(X,J+2
     ):Y2=N(X,J+3)
3170 IF X1<0 THEN 3185
3175 PLOT X1+X7,Y1+Y7:DRAWTO X2+X7,Y2
     +Y7
3180 NEXT J
3185 C5=0:RETURN
3190 POSITION 1,1:GOSUB 3060:RETURN
3500 IF A$="/" THEN 510
3510 RETURN
```

©

Learning With Computers

Glenn M. Kleiman

Computerized Drill And Practice

There is a very old joke which starts with the question: How does one get to Carnegie Hall from here? The answer is, of course: practice, practice, practice.

Practice is necessary to become proficient at any skill, whether it is a musical skill such as playing the piano, a physical skill such as riding a bicycle, or the more cognitive skills of reading, writing and arithmetic. In each case, beginners must concentrate their effort and attention on basic components of the skill. Beginning pianists think about the location of each note, beginning bicyclists attend to balancing, steering and pedaling, and beginning readers concentrate on recognizing each word.

After extensive practice, the individual becomes agile and can perform the basics without much effort or attention. Proficient pianists move their fingers almost automatically, and can therefore concentrate on the music, not the physical actions of playing the notes. After practice, bicyclists can balance, steer and pedal without attending to their movements. Proficient readers recognize most words quickly and effortlessly, and therefore can focus their attention on the overall meaning of the text.

Rote Drills And Practice

Despite the obvious need and value of practice, there are controversies about the drill and practice work that occupy so much of students' and teachers' time. One controversy centers on the amount of time and effort which should be devoted to drill work, as opposed to more conceptual, exploratory, or creative endeavors. Another controversy centers on the nature of the practice exercises given to students.

Many educators believe that common approaches to reading, math, and other drills are not effective and, in some cases, may even be detrimental. Often, this debate is over the virtues of dividing skills into many subskills and having students practice each one in isolation, as opposed to practicing the entire skill at once. The most common example is in the teaching of reading, where the contrast is between emphasis on practicing phonics and word recognition subskills versus emphasis on practicing reading real books, magazines, and newspapers.

The introduction of computers into schools has involved these debates about drill and practice. Drill work was the first use of computers in many schools, and it continues to be a prevalent application. There is more software for math and other drills than for any other educational application of computers. However, many educators decry such use of computers. They strongly advocate that the limited number of computers in schools be used to encourage conceptual learning, not rote drills.

I concur, to a large extent, with those who criticize the drill and practice exercises so common in many schools. However, I do not agree that computers should never be used for drill and practice. Practice is, I think, a necessary evil, one which is essential for mastering any skill. Computers, with properly designed software, can make the practicing of certain skills both more effective and more enjoyable.

Many types of practice follow a similar format. Practice items, such as math problems or typing drills, are presented to the students. The students respond to each item, answering questions or performing actions such as typing sequences of letters. At some point, the students receive feedback on their work. In many skills, speed as well as correctness is important, so the feedback covers both. Students are then expected to direct further study and practice to those items with which they had difficulty.

Effective Computer Exercises

Several factors determine the effectiveness of practice drills. First, the selection of the practice items is critical. There is no value to practicing already mastered items, and items that are too difficult will lead to frustration rather than learning. Certain characteristics of feedback are also critical.

Immediate feedback is much more valuable than delayed feedback, since it enables students to catch their errors and learn the correct response while they are still actively involved in the drill. Immediate feedback also helps keep students' attention on their work.

Also important is whether the feedback helps students understand and correct their errors. Feedback that explains why responses are incorrect leads to much more effective learning than feedback which simply tells students whether their answers are correct or incorrect.

Computers can be programmed to present practice items, monitor students' performance, adjust the items to an appropriate level for each individual, and provide immediate and, in many cases, explanatory feedback. For skills in which speed is important, computers can accurately measure the time of every response and control how quickly practice items are presented.

Learning to type provides a good example of the possible benefits of using computers. Everyone agrees that typing is a valuable skill, one that is becoming even more valuable as computers are used more widely. The only way to become a proficient typist is through repetitive practice. Computers can make practice more effective, so less time need be devoted to it. Computers can also free teachers from the drudgery of correcting typing tests.

Several companies market programs to help people learn to type. When these programs are used, the computer presents sequences of letters and words on the screen, and the student types them. The drills follow established methods of teaching typing, so they begin with the "home" keys (ASDFJKL;) and then gradually add other letters. As the student types each sequence, the computer monitors both accuracy and speed. It can make students immediately aware of their errors, so that incorrect habits do not become ingrained.

In addition, the computer can identify keys and sequences on which the student needs to gain more speed. The programs automatically adjust later drills so that practice time is directed to those letters and sequences that are most in need of further work. This continuous dynamic adjustment of the drill items can be accomplished only with computers.

Practice With Games

Computers can also make drills more enjoyable by incorporating them into games. In one such program, called *MasterType* (from Lightning Software, P.O. Box 11725, Palo Alto, CA 94306), typing drills are placed into the context of a space invaders game. The scenario has the player defending his planet against attackers from the planet Lexicon. The attackers are represented by letters or words in each of the four corners of the screen. The attackers fire missiles at the planet. The player must destroy the attackers by quickly and accurately typing each of the words. The excellent arcade-like features have many people so caught up in the game they forget they are actually involved in the drudgery of typing practice. The same approach is used in a series of well-designed programs from Developmental Learning Materials, Inc. (1 DLM Park, Allen, TX 75002) which incorporate math drills into arcade-like games.

Music training is another area in which computerized drills can be beneficial. Several music drill programs are available. The following examples are based on programs developed by the Minnesota Educational Computing Consortium.

One drill helps train students to recognize and produce rhythms. The computer presents a sequence of notes on the screen. The student is asked to tap the rhythm by pressing the space bar on the computer keyboard. The computer immediately checks the answer. When the student makes an error, the computer plays the original rhythm and the one tapped out by the student, thereby aiding understanding. Another part of the drill plays rhythms and has the student specify the length of each note.

Another music drill helps students learn to recognize musical notes and musical notation. In this drill, the student sees written notes on the computer screen and then hears notes played. In each case, one of the notes played does not match the corresponding note in the written sequence. The student's job is to find the incorrect note. If the student makes a mistake, the computer repeats the original sequence and plays the notes as written, so that the student can hear the difference.

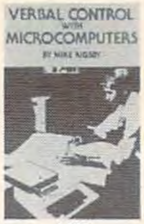
These are but a few examples of the potential benefits of computerized drill and practice. These benefits can, of course, also be applied to more academic skills, such as math and spelling drills. A great deal of drill and practice software is available. This software varies in how well it takes advantage of the potential value of computer-assisted practice. Careful evaluation is necessary before selecting any drill and practice program. ©

The Computer Book Club®

"The ONLY Book Club for micro users"

Play games . . . develop new programming skills . . . organize your finances . . . earn extra income . . . even build your own microcomputer . . .

1480
List $22.95



1468
List $18.95

1394
List $15.95

HAVE IT ALL FOR ONLY $2.95

Select any 5 books, pay only $2.95

(a value up to $102.75) **when you join The Computer Book Club®!**

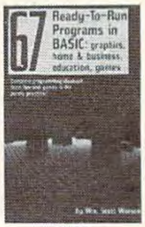
1160
List $13.95

1398
List $16.95

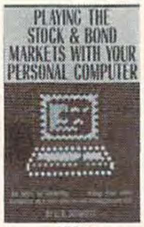
1299
List $16.95

1391
List $17.95

337
List $19.95



1195
List $13.95


PLAYING THE STOCK & BOND MARKETS WITH YOUR PERSONAL COMPUTER

1251
List $16.95

1199
List $18.95

841
List $6.95 (paper)

1050
List $12.95

1276
List $15.95

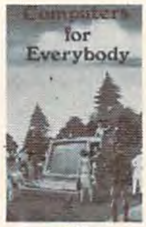
1396
List $13.95

1369
List $21.95

HOW TO DESIGN, BUILD & PROGRAM YOUR OWN WORKING COMPUTER SYSTEM

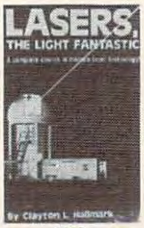
1111
List $15.95

1293
List $15.95



338
List $14.95

1045
List $13.95



1108
List $14.95

1423
List $17.95

1496
List $17.95

1169
List $17.95

1205
List $16.95

1085
List $14.95

1406
List $17.95

1183
List $14.95

1295
List $16.95

1506
List $12.95

7 very good reasons to try The Computer Book Club®

Blue Ridge Summit, PA 17214

- **Reduced Member Prices.** Save 20% to 75% on books sure to increase your computer know-how
- **Satisfaction Guaranteed.** All books returnable within 10 days without obligation
- **Club News Bulletins.** All about current selections—mains, alternates, extras—plus bonus offers. Comes 13 times a year with dozens of up-to-the-minute titles you can pick from
- **"Automatic Order."** Do nothing, and the Main selection will be shipped automatically! But . . . if you want an Alternate Selection—or no books at all—we'll follow the instructions you give on the reply form provided with every News Bulletin
- **Continuing Benefits.** Get a Dividend Certificate with *every* book purchased after fulfilling membership obligation, and qualify for discounts on many other volumes
- **Extra Bonuses.** Take advantage of added-value promotions, plus special discounts on software, games, and more
- **Exceptional Quality.** All books are first-rate publisher's editions, filled with useful, up-to-the-minute information

The Computer Book Club®

Blue Ridge Summit, PA 17214

Please accept my membership in The Computer Book Club and send the 5 volumes circled below, billing me $2.95 plus shipping and handling charges. If not satisfied, I may return the books within ten days without obligation and have my membership cancelled. I agree to purchase 4 or more books at reduced Club prices (plus shipping/handling) during the next 12 months, and may resign any time thereafter.

337 338 841 1045 1050 1085 1108
1111 1160 1169 1183 1195 1199 1205 1251
1276 1293 1295 1299 1369 1391 1394
1396 1398 1406 1423 1468 1480 1496 1506

Name ______________________ Phone ________

Address ______________________________

City _________________________________

State ______________________ Zip ________

(Valid for new members only. Foreign and Canada add 20%. Orders outside U.S. or Canada must be prepaid with international money orders in U.S. dollars). This order subject to acceptance by The Computer Book Club. CMPT-383

FIGHTER ACES-
Add A Second VIC Joystick

John Parr

This game, Fighter Aces, is fun in its own right. But it also shows a simple way to add a second joystick to your VIC for two-player games.

I spend many hours behind the CRT on my VIC, attempting one program or another, but when the work is done, I am not ashamed to play a game or two for relaxation. Many of the games that I like, however, require two joysticks.

Other programmers have circumvented this problem through the use of keys, but I find the use of keys awkward. Besides, most games use the same keys over and over, which I am sure must be wearing on my precious investment. The only answer to my dilemma, therefore, was to find some way of connecting a second joystick.

Before I went to work, I decided that I'd better find out a little bit about how the joysticks worked. As it turns out, the VIC joystick is just a lever connected to four micro switches at its base. When the stick is pressed in one direction, the lever closes the appropriate switch, grounding one of the pins on the games port. For diagonals, two switches are closed simultaneously, grounding two pins in the games port. When a pin is grounded, one bit is turned off in either memory location 37137 or in location 37152. (For any who do not know what a "bit" is, I refer you to **COMPUTE!**, November and December 1981, #'s 18 and 19, "An Introduction to Binary Numbers.")

From this understanding, I decided that the best place to hook a second joystick on was through the parallel user port. (As it turns out, PET users have been doing this for years.) After a little checking of my memory map, I decided to connect my second joystick on pins D through J, grounding to pin A. These pins are easily read through memory location 37136.

My next chore was to determine the most logical order in which to make my connections. I finally decided on a system by which any formulas for the first joystick could be used by the second. The following hookup is the result of my research.

Looking at the plug on the joystick, you will see this (minus the numbers, of course):

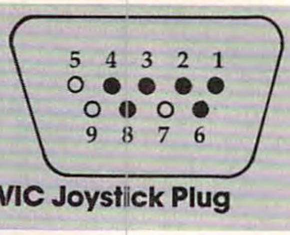


Figure 1: VIC Joystick Plug

The filled-in holes represent pins which are used. You will notice that this is a mirror image to the diagram which is in your VIC book.

The following chart tells what each pin does:

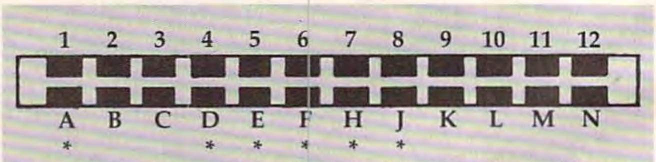
Table.

Pin number	Description	
1	Up	– Joy 0
2	Down	– Joy 1
3	Left	– Joy 2
4	Right	– Joy 3
6	Fire Button	
8	Ground	

Simply connect these pins to a 24-pin edge connector as follows:

Joystick		Edge Connector
1	to	E
2	to	F
3	to	H
4	to	D
6	to	J
8	to	A

The 24-pin edge connector then plugs into the User I/O Port on the back of the VIC, which has the configuration shown in Figure 2.



Pins to which connections are made are marked with an asterisk(*).

Figure 2: VIC User I/O Port

These connections can be made either by replacing the existing joystick plug with the edge

Cardco, Inc. announces five All-American ways to . . .

Expand your VIC® at affordable prices

card? A

A universal centronics parallel printer interface for the VIC-20 & C-64 computers. Obeys all standard VIC print commands.
Suggested Retail — $79.95

```
**** CBM BASIC V2 ****
28023 BYTES FREE
READY.

        CARDCO, INC.
        CARDCO, INC.
        CARDCO, INC.
        CARDCO, INC.
```

cardette 1

A universal cassette interface for the VIC-20 & C-64 computers. Emulates all functions of the data cassette.
Suggested Retail — $39.95

cardboard 3

The CARDBOARD 3 is a fuse protected, economy expansion interface designed to allow the user to access more than one of the plug-in-type memory or utility cartridges now available. It will accept up to three cartridges at once. This product includes reset button and switches.
Suggested Retail — $39.95

cardriter 1

A light pen for the VIC-20 and C-64 computers with a switch on the barrel and 6 good programs.
Suggested Retail — $39.95

cardboard 6

The CARDBOARD 6 is a fuse protected expansion interface designed to allow the user to access more than one of the plug-in-type memory or utility cartridges now available. Additionally it allows switch selection of games and other programs now available in the cartridge format, without the necessity of turning the computer off and on again, thereby saving a great deal of stress on your VIC-20 and on your television or monitor.
Suggested Retail — $99.95

All Cardco products are **Made in the U.S.A.** and are individually tested to ensure quality and reliability. Superior technological engineering optimizes the value/performance ratio of all of our products.

Specifications and prices subject to change.

Dealer inquiries invited.
United States: Cardco, Inc. • 313 Mathewson • Wichita, KS 67214 • (316) 267-6525
West Canada: LSI Distributing • Attn: Mr. Wong • 2091 W. 61st Avenue • Vancouver, BC. CA V6J 1Z2 • (604) 733-0211
England & Europe: Audiogenic • Martin Manary • 34-36 Crown St. • Reading, Berkshire England • (0734) 595647
East Canada: Hobby Craft Canada • 24 Ronson Drive • Rexdols Ontario M9W1B4 • (416) 241-2661

®VIC-20 is a registered trademark of Commodore connector or by using a "patch" cord. A "patch" cord is an extension cord with one type of plug on one end (such as our edge connector and with a different type of plug on the other end (such as a connector like the one which is mounted on the side of your computer for the games port). I personally prefer the "patch" cord method, because then the joysticks can be interchanged if one wears more than the other. Also, if a joystick breaks for some reason, there need be no changes made to the replacement.

From the arrangement I have chosen, all formulas used on one joystick can be used on the other with little modification. Personally, I find the new joystick easier to use because all switches can be read from the same memory location with one simple PEEK. I like it so much, in fact, that it has become my main joystick.

Fighter Aces

Now that I have shown you how to connect this joystick, I will show you how to use it with one of my favorite games, Fighter Aces. In this game, two players engage in a "dog-fight" across your VIC's screen. The game ends at fifteen points; may the best pilot win!

As it is written, this program will run on any memory configuration the VIC can attain.

Before continuing, I must explain the use of the decimal points. The decimal point is a constant for the number zero. The only difference between the use of the decimal point and the use of a zero is that decimal points will speed program execution. If you feel ambitious, try replacing the decimals with zeros to see what I mean.

At last, we have arrived at our program description. The code follows a fairly simple algorithm, so with the explanation, you should be able to understand its workings.

Lines	Description
10-50	Set the program to run with any memory by changing the locations of the screen and color. Also, these lines move the variable storage above the user-defined characters if your computer is expanded by 8K or more; if not, the program sets the end of memory below the special characters, thus protecting them for any memory configuration.
60-150	Set up the variables and the screen before the game begins.
160-170	Get values for each joystick.
180-220	Check for a fire button; see if a shot has already been fired. Each shot is checked here to see if it has gone to the end of its limited range. Note: By eliminating line 180 and the NEXT on line 290, the biplanes will be more responsive, but the shots will be slower. Conversely, if the value of the loop is upped, the shots will move faster, but the planes will be harder to control.
230-280	Move the shots checking for out of bounds, out of range, and a hit.
290-340	Set new direction on each biplane and determine which type of biplane is to be POKEd.
350-400	Move each biplane, checking for out of bounds and crashes.
410-440	Subroutine to determine what a shot hit. (Control tower, another shot, or a biplane.)
450-540	Subroutine for an explosion. Also checks for a mid air collision and updates the score. If either score equals fifteen, the ending flag(s) are set.
550-650	Game over routine.
660-790	Create the biplanes and print the title page.

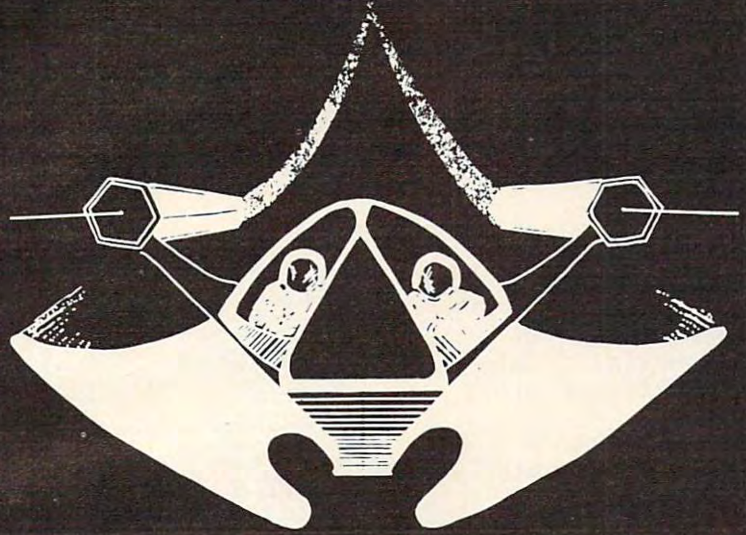
Important Variables:

S	The first sound channel.
V%	The starting address of the video display.
C	The difference between the screen and color locations.
P%()	Position of each plane on the screen.
SP%()	Position on the screen of each shot.
SD%()	Direction of each shot.
SF%()	Flag to show whether a shot is on the screen and, if it is, how far it has to travel.
D%()	Direction of each plane.
A%()	The attitude of each plane.
SC%()	The score for each player.
E%()	Flag to show if someone has fifteen points.
G%()	The number of games that each player has won.
M%()	Value from each joystick.
L%	Flag for the biplane being out of screen limits.

```
10 IFFRE(Ø)>7ØØØTHENPOKE46,32:GOTO3Ø
20 POKE56,29
3Ø CLR:S=36874:POKE4+S,5:POKE36879,25
4Ø V%=4*(PEEK(36866)AND128)+64*(PEEK(3686
   9)AND12Ø):C=37888+4*(PEEK(36866)A
   ND128)-V%
5Ø GOTO66Ø
6Ø DIMP%(1),SP%(1),SD%(1),SF%(1),D%(1),A%
   (1),SC%(1),E%(1),G%(1)
7Ø DEFFNM(X)=((XAND4)=.)*22+((XAND16)=.)-
   ((XAND2)=.)-((XAND8)=.)*22
8Ø GOTO12Ø
9Ø P%(.)=V%+463:A%(.)=.:D%(.)=1:RETURN
1ØØ P%(1)=V%+482:A%(1)=4:D%(1)=-1:RETURN
11Ø PRINT"{HOME}{CYN}{REV}SCORE:":PRINTTAB
    (5)"{REV}{BLK}"SC%(.)TAB(14)"{WHT
    WHT}"SC%(1):RETURN
12Ø PRINT"{CLEAR}{GRN}{Ø2 DOWN}{REV}******
    ****************":FORX=1TO18:PRIN
    T:NEXT:PRINT"{REV}{CYN} #########
    #";
13Ø PRINT"{UP}B{UP}{LEFT}B{UP}{LEFT}B{UP}{
    LEFT}B{Ø4 DOWN}{LEFT}##########{HOME}"
14Ø GOSUB9Ø:GOSUB1ØØ:GOSUB11Ø
15Ø POKES+3,2ØØ:POKES,2ØØ
16Ø POKE37154,127:X=PEEK(37152):POKE37154,
    255:M%(1)=2*(X=119)+PEEK(37137)
17Ø M%(.)=PEEK(37136)-129
18Ø FORY=1TO2
19Ø FORX=.TO1:IFM%(X)AND32THENNEXT:GOTO23Ø

2ØØ IFSF%(X)THENNEXT:GOTO23Ø
21Ø SF%(X)=11:SP%(X)=P%(X)+D%(X):SD%(X)=D%
    (X)
22Ø IFSP%(X)>V%+483ORSP%(X)<V%+66ORPEEK(SP
```

GAME PROGRAM DEVELOPMENT KIT



for the

VIC - 20 and COMMODORE - 64

Registered trademarks 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!!!

DESIGNED TO RUN ON ALL VIC-20's

$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

```
    %(X))=194THENSF%(X)=.:NEXT:GOTO23
    Ø
230 FORX=.TO1:IFSF%(X)=.THENNEXT:GOTO29Ø
240 SF%(X)=SF%(X)-1:IFSF%(X)=.THENPOKESP%(
    X),32:NEXT:GOTO29Ø
250 POKESP%(X),32:SP%(X)=SP%(X)+SD%(X)
260 IFSP%(X)<V%+66ORSP%(X)>V%+483THENSF%(X
    )=.:NEXT:GOTO29Ø
270 IFPEEK(SP%(X))<>32THENSF%(X)=.:GOTO41Ø
    :NEXT:GOTO29Ø
280 POKESP%(X)+C,X:POKESP%(X),41:NEXT
290 NEXT:FORX=.TO1:IF(M%(X)AND3Ø)=3ØTHEN35
    Ø
300 D%=FNM(M%(X)):IFD%=D%(X)THEN35Ø
310 D%(X)=D%:A=(D%/11):IFA>2THENA=A+1
320 IFA<-2THENA=A-1
330 IFA<.THENA=A-4
340 A%(X)=ABS(A)
350 IFP%(X)+D%(X)<V%+66ORP%(X)+D%(X)>V%+48
    3THENC%=X:L%=1:GOSUB45Ø
360 IFPEEK(P%(X)+D%(X))<>32THENC%=X:GOSUB4
    5Ø
370 IFE%(.)ORE%(1)THEN55Ø
380 POKEP%(X),32:P%(X)=P%(X)+D%(X)
390 POKEP%(X)+C,X:POKEP%(X),A%(X)+33
400 NEXT:GOTO16Ø
410 IFPEEK(SP%(X))=194THEN29Ø
420 IFPEEK(SP%(X))=41THENPOKESP%(X),32:SF%
    (.)=.:SF%(1)=.:GOTO29Ø
430 C%=1-X:GOSUB45Ø:IFE%(X)THEN55Ø
440 GOTO29Ø
450 POKEP%(C%),42:POKES+4,15:FORI=1TO7Ø:NE
    XT:POKES+4,5:POKEP%(C%),32
460 H%=PEEK(P%(C%)+D%(C%))
470 IFH%=41THENSF%(1-C%)=.:POKEP%(C%)+D%(C
    %),32:H%=32
480 IFH%<>32ANDH%<>194THENB%=1
490 SC%(1-C%)=SC%(1-C%)+1
500 IFSC%(1-C%)=15THENE%(1-C%)=1
510 ONC%+1GOSUB9Ø,1ØØ
520 IFL%THENL%=.:B%=.:GOTO54Ø
530 IFB%THENB%=.:C%=1-C%:GOTO45Ø
540 GOSUB11Ø:RETURN
550 POKES+4,Ø
560 IFE%(.)ANDE%(1)THENPRINT"{CLEAR}{REV}T
    IE GAME !!":GOTO6ØØ
570 W%=-(E%(.)=1)-2*(E%(1)=1)
580 PRINT"{CLEAR}{REV}PLAYER"W%" WINS."
590 G%(W%-1)=G%(W%-1)+1
600 PRINT"{02 DOWN}{CYN}{REV}* CURRENT  ST
    ANDINGS *":FORX=.TO1:PRINT"{DOWN}
    {YEL}{REV}PLAYER"X+1" -"G%(X):NEX
    T
610 PRINT:PRINT"{BLK}{REV}PLAY AGAIN?"
620 GETA$:IFA$=""THEN62Ø
630 IFA$<>"N"THENSC%(.)=.:SC%(1)=.:E%(.)=.
    :E%(1)=.:POKES+4,5:GOTO12Ø
640 PRINT"{CLEAR}{BLU}"
650 POKE36869,24Ø+48*(V%=4Ø96):FORX=.TO4:P
    OKES+X,Ø:NEXT:POKE36879,27:END
660 PRINT"{CLEAR}{BLU}{DOWN}* * FIGHTER  A
    CES! * *"
670 FORX=1TO5:PRINT:NEXT
680 PRINTTAB(7)"{BLK}ANOTHER":PRINT:PRINTT
    AB(9)"JHP":PRINTTAB(9)"VIC":PRINT

690 PRINTTAB(7)"PROGRAM"
700 FORX=.TO1Ø:READY:FORZ=.TO7:READA:POKEZ
    +Y,A:NEXT:NEXT:
710 DATA7464,Ø,56,145,187,255,187,145,56,7
    44Ø,4,22,39,88,58,18Ø,72,32
720 DATA7448,6Ø,24,Ø,9Ø,126,9Ø,Ø,6Ø,7456,3
    2,1Ø4,228,26,92,45,18,4
730 DATA7432,Ø,28,137,221,255,221,137,28,7
    472,4,18,45,92,26,228,1Ø4,32
740 DATA748Ø,6Ø,Ø,9Ø,126,9Ø,Ø,24,6Ø,7488,3
    2,72,18Ø,58,88,39,22,4,7496,Ø,Ø,Ø
    ,24,24,Ø,Ø,
750 DATA75Ø4,153,9Ø,6Ø,255,255,6Ø,9Ø,153,7
    424,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø
760 FORX=1TO6:PRINT:NEXT
770 PRINT"{GRN}PRESS RETURN TO BEGIN{HOME}
    "
780 GETA$:IFA$<>CHR$(13)THEN78Ø
790 PRINT"{CLEAR}":POKE36869,255+48*(V%=4Ø
    96):POKE36879,11Ø:GOTO6Ø
```

©

NEW
VIC SOFTWARE VIC

Great VIC Software

Use Joystick or keyboard

ALIEN INVASION — Arcade style excitement for your VIC. Look out here they come. Aliens are descending from the sky. Move your laser into position and defend the earth. The attacks are unending — can you survive or will Vader rule the galaxy. Many extras on this one. 20 levels of play. $12.95

CATTLE-ROUNDUP — The cows are loose in the maze. You have 2 minutes to get each cow back into the corral. You can push, coax and call the cows. Some cows are not very smart and some are very stubborn. You will have to help them. Be careful that you don't leave the corral gate open. Color graphics and sound. Eight levels of play and a time limit. $12.95

HEAD ON — Your car moves forward around the race track. You can move up, down, right and left. Try to score points by running over the dots on the track. Watch out for the crusher — if you crash you lose a car. Four cars and bonus levels. Full color graphics and sound. Fast action and very addicting. 9 levels of play. $12.95

SNAKEOUT — Blocks appear on the screen at random. You move up, down, right and left and try to move your snake over the blocks. Each block that you get raises your score. Keep building your score but watch out because the escape routes keep getting smaller. Time limit, color graphics and sound. 3 games on this cassette. Snakeout — 2 player Snakeout and Trapper. 9 Levels of Play. $12.95

TARGET COMMAND — Move your laser into position and get ready for some quick action. Different types of missiles are dropping. How many can you shoot down. They all travel at different speeds and different levels. You must be fast on the trigger to get them all. Time limit, bonus points and very addicting. Color graphics and sound. Arcade style fun. 10 levels. $12.95

COMMODORE 64 SOFTWARE

Let the ELECTRIC COMPANY turn your 64 into a home arcade!

COLOR • GRAPHICS • SOUND
ON CASSETTE

ARCADE PAK - $24.95
3 Programs
Head On
Alien Invasion
Target Command

EDUCATION PAK $24.95
3 Programs
Geography Match
Math - Adventure
King

ADVENTURE PAK - $14.95
2 Programs
Adventure
Caves of Silver

GAME PAK $14.95
2 Programs
Dragon Chase
Deflect

Joystick and Keyboard versions included

COMPUTERMAT • BOX 1664, DEPT **-20**
LAKE HAVASU CITY, ARIZONA 86403

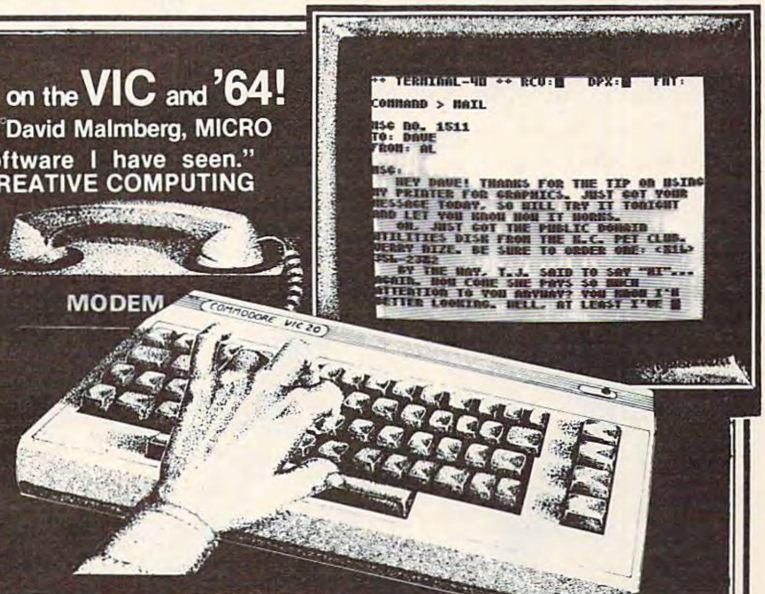
WRITE FOR
FREE CATALOG

THE ELECTRIC COMPANY
P.O. Box 388C • Lake Havasu City • Arizona 86403

TELECOMMUNICATIONS on the VIC and '64!

"A versatile and exceedingly well-done package." David Malmberg, MICRO

"Simply the best & nicest VIC terminal software I have seen."
Greg Yob, CREATIVE COMPUTING



```
** TERMINAL-40 ** RCV:█  DPX:█  FMT:

COMMAND > MAIL

MSG NO. 1511
TO: DAVE
FROM: AL

MSG:
  HEY DAVE! THANKS FOR THE TIP ON USING
Y PRINTER FOR GRAPHICS. JUST GOT YOUR
ESSAGE TODAY. SO WILL TRY IT TONIGHT
ND LET YOU KNOW HOW IT WORKS.
  OH, JUST GOT THE PUBLIC DOMAIN
TILITIES DISK FROM THE K.C. PET CLUB.
ERRY NIZE. BE SURE TO ORDER ONE: <816>
[illegible]-2382
  BY THE WAY, T.J. SAID TO SAY "HI"...
GAIN. HOW COME SHE PAYS SO MUCH
TTENTION TO YOU ANYWAY? YOU KNOW I'M
ETTER LOOKING. WELL, AT LEAST I'VE █
```

We created quite a flurry and earned *rave* reviews with *Terminal-40*, the unique software that transforms the VIC screen into a *40-column smooth-scrolling* display. And with features like a Receive Buffer and VIC printer dump, *Terminal-40* sets a new standard for personal modem communications with networks such as CompuServe and Source. Our *'64 Terminal* does the same quality job for the '64.

And now there's even MORE! ! ! *SuperTerm-40* and *SuperTerm '64* support text storage to disk or tape and program UPLOAD/DOWNLOAD. SuperTerms, used with our Smart ASCII, also support popular parallel printers.

Choose the one right for you. Call or write today for the "best", then . . .

REACH OUT and BYTE SOMEONE!

For the VIC:
Terminal-40** (req 8K exp)* . . ***$29.95
SuperTerm-40** (req 16K exp)* . . ***Call
For the Commodore 64:
'64 Terminal** **$29.95
SuperTerm '64** . . . . **Call
(On cassette. Requires modem; VIC printer optional.)

MIDWEST MICRO associates
PO BOX 6148, KANSAS CITY, MO 64110

ORDER DESK: (9 am - 4 pm. Orders only!)
(816) 254-9600
Technical support (816) 921-6502
Send for a free brochure describing our other quality products.

MAIL ORDER: Add $1.25 shipping and handling ($3.50 for C.O.D.); VISA/Mastercard add 3% (card# and exp. date). Missouri residents include 4.6% sales tax. Foreign orders payable U.S.$, U.S. Bank ONLY; add $5 shp/hndlg. **Dealer inquiries invited.**

TI Trapshoot

C. Regena

It's time to try your skill at the TI-99/4A trapshooting range. Each game consists of 50 clay pigeons sprung from the trap near the center of the screen. You aim the shotgun located at the bottom center of the screen by pressing the left or right arrow key. Fire when you are ready by pressing the ENTER key. You'll need to aim and fire quickly – you have only one chance per target.

At the lower right side of the screen is your scoring record. The number of successful hits, the number of clay pigeons, and the number of rounds fired are displayed.

Programming Techniques

Line 120 defines a function RRV for the random row velocity for the clay pigeon moving from 0 to 140 upward. You may change the number 14 in the equation to 15 or 16 to make the target move upward more quickly, but you will have less time to aim the shotgun, shoot, and hit the target.

Line 130 defines a function RCV for the random column velocity of -17 to +17 moving the target toward the left or right. The number 18 in the equation may be changed to decrease or increase the range of the target. Increasing the number will move the target more to the left or right, but the target may "wrap" to the other side of the screen before being deleted.

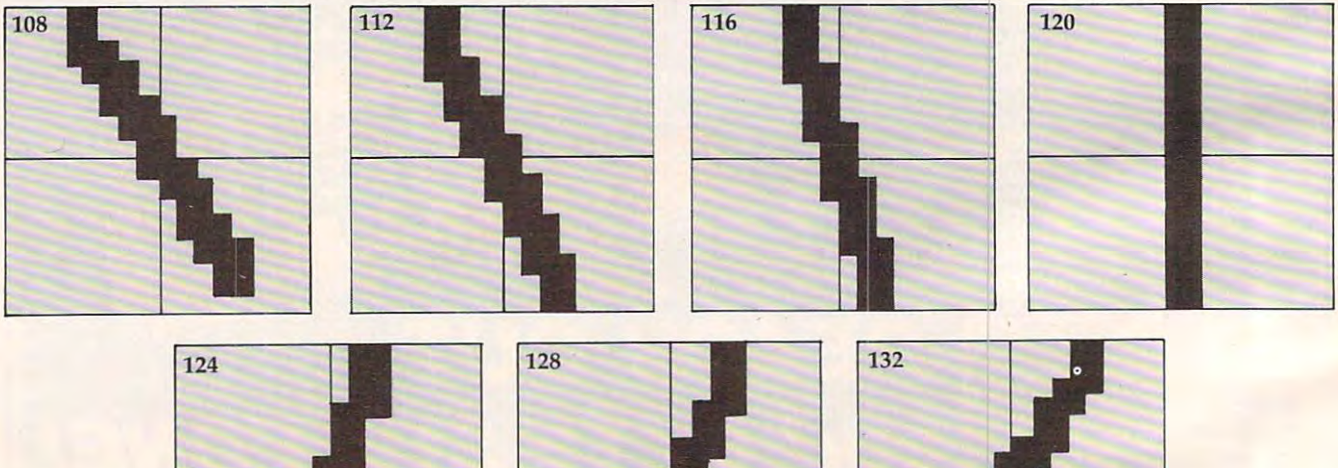
The shotgun is Sprite #3, defined in Line 190. There are seven shotgun positions drawn with characters 108 through 135. CALL MAGNIFY(4) is used so the shotgun may be drawn as large as possible by specifying only one character number for the sprite. If the left arrow key is pressed, the character number N is decreased by 4; if the right arrow key is pressed, N is increased by 4. N may vary from 108 to 132, where 120 is straight up. The shotgun position is changed after a key is pressed by using CALL PATTERN(#3,N).

Line 220 stops the game after 50 clay pigeons. You may change the limits of the game by changing the limit for T, or you may wish to test for the number of shots, SH (perhaps stopping after 50 rounds or 100 rounds instead of after 50 birds).

Line 230 springs the clay pigeon from the trap at the random row and column velocities. The target is Sprite #1.

If you press ENTER to fire, buckshot appears as Sprite #2 at the end of the shotgun and goes upward in the direction the shotgun is aimed. The value of N2 is N-120 and is used in calculating position and column velocity parameters for Sprite #2. The position of the end of the shotgun is dot column 116 plus some function of N2. Experimentation shows that the dot column position is 116+N2*1.2.

Following are the graphic representations of the seven positions of the shotgun:



By using trigonometry, the angle of the shotgun was determined dependent upon the character number. The ratio of the row velocity to the column velocity is equal to the ratio of the horizontal displacement to the vertical length of the shotgun. Whether the shotgun is pointing left or right is determined by SGN(N2). The upward (row) velocity of Sprite #2 was set at 100.

The theoretical factor to calculate column velocity is 12.5, but since the displacement per character number is not precisely linear, 12.7 works better. The resultant column velocity is (N2/4 + 2*SGN(N2))*12.7. I chose the row velocity of 100 so the buckshot moves faster than the clay pigeon, but slowly enough to report coincidence and to prevent wrapping on the screen.

Controlling Sprites

Lines 330-340 check to see if the buckshot hits the target. CALL COINC(ALL,C) is used so coincidence is reported if any dot of the buckshot coincides with any dot of the target. Using a statement such as CALL COINC(#1,#2, TOL,C) between two sprites tests coincidence of the upper left corners of each sprite within a certain tolerance; sometimes a hit would be scored when the buckshot appeared to miss the target.

The faster sprites move, the more difficult it is to control them in a program. Coincidence is reported only if the sprites are touching at the exact moment the CALL COINC statement is executed in the program. Once ENTER is pressed and the buckshot starts on its path, CALL COINC is executed in a FOR/NEXT loop 19 times. At the end of 19 loops without coincidence, the buckshot is near the top of the screen and is deleted.

If coincidence is reported, then the program branches to the appropriate section for a hit. If you change the speed of either the target or the buckshot, you may need to change the limit 19 in the FOR/NEXT loop. If you play many times, you may notice that once in a while the buckshot will pass through the target without recording a hit. This happens when the target is going straight upward slowly and you fire immediately. The sprites pass each other before the program has a chance to get to the CALL COINC statement. To avoid this problem, you could slow the buckshot down; however, I prefer the faster buckshot since the problem rarely occurs. This is an example of "programming trade-offs."

After the buckshot is deleted, the program keeps testing the position of the target until it is at the top of the screen; then Sprite #1, the target, is deleted (line 360). If ENTER is not pressed, then the position of Sprite #1 is tested in the CALL KEY loop.

If the target is hit, then the broken clay pigeon is shown by changing the pattern of the sprite. The buckshot disappears by changing the pattern of the buckshot to a blank character. The statement is CALL PATTERN(#1,100,#2,136). After sounding a hit using Noise -6, both sprites are deleted with CALL DELSPRITE(#1,#2).

99/4 Versus 99/4A

Note: Some of the consoles process at different rates. It makes a difference whether you have the TI-99/4, the earlier TI-99/4A, or the later TI-99/4A. It also makes a difference if you have the old Extended BASIC module or the new Extended BASIC. (You can tell which you have by holding a key down. If it will automatically repeat, you have a newer module.) Since this game is very critical on timing, you will have to experiment a little so that sprites won't wrap and cause bugs. You can adjust the game by changing the limit in line 330.

New XBASIC, TI-99/4A	330 FOR I = 1 TO 19	(or 20)
Old XBASIC, TI-99/4A	330 FOR I = 1 TO 9	
New XBASIC, TI-99/4	330 FOR I = 1 TO 19	
Old XBASIC, TI-99/4	330 FOR I = 1 TO 12	

If you prefer to save your typing time and effort, I will send you a copy of this program if you send me $3 plus a stamped, self-addressed mailer, and a blank tape or disk.

C. Regena
P.O. Box 1502
Cedar City, Utah 84720

```
100 REM TRAPSHOOT
110 REM TI EXTENDED BASIC
120 DEF RRV=-INT(RND*10+14)
130 DEF RCV=(-1)^(INT(RND*4+1))*(INT(
    RND*18))
140 GOTO 460
150 RANDOMIZE :: CALL CLEAR :: N=120
    :: H,T,SH=0
160 CALL COLOR(8,3,1):: CALL HCHAR(24
    ,1,92,32)
170 CALL HCHAR(15,14,140):: CALL HCHA
    R(15,15,141,3):: CALL HCHAR(15,18
    ,142)
180 CALL HCHAR(16,14,141,5):: CALL HC
    HAR(17,14,141,5)
190 CALL SPRITE(#3,N,5,160,108)
200 DISPLAY AT(21,19):"HITS:" :: DISP
    LAY AT(22,19):"BIRDS:" :: DISPLAY
    AT(23,19):"ROUNDS:"
210 DISPLAY AT(21,26):USING "###":H :
    : DISPLAY AT(22,26):USING "###":T
    :: DISPLAY AT(23,26):USING "###":SH
220 IF T=50 THEN 410
230 T=T+1 :: CALL SPRITE(#1,96,7,112,
    117,RRV,RCV):: CALL SOUND(150,-5,
    0)
240 CALL KEY(0,KEY,S)
250 IF KEY=13 THEN 310
260 IF KEY<>83 THEN 280 ELSE N=N-4 ::
     IF N<108 THEN N=108
270 CALL PATTERN(#3,N):: GOTO 300
280 IF KEY<>68 THEN 300 ELSE N=N+4 ::
     IF N>132 THEN N=132
290 CALL PATTERN(#3,N)
300 CALL POSITION(#1,R,C):: IF R>10 A
    ND R<112 THEN 240 ELSE CALL DELSP
    RITE(#1)::GOTO 210
```

```
310 CALL SOUND(1000,-4,0):: N2=N-120
320 CALL SPRITE(#2,104,2,154,116+N2*1
    .2,-100,(N2/4+2*SGN(N2))*12.7)
330 FOR I=1 TO 19 :: CALL COINC(ALL,C
    ):: IF C=-1 THEN 370
340 NEXT I
350 CALL DELSPRITE(#2)
360 CALL POSITION(#1,R,C):: IF R>5 AN
    D R<112 THEN 360 ELSE CALL DELSPR
    ITE(#1)::SH=SH+1 :: GOTO 210
370 CALL PATTERN(#1,100,#2,136):: CAL
    L SOUND(1000,-6,0)
380 CALL DELSPRITE(#1,#2)
390 SH=SH+1 :: H=H+1
400 CALL SOUND(1,-6,30):: GOTO 210
410 CALL DELSPRITE(ALL):: CALL HCHAR(
    24,1,32,32):: CALL COLOR(8,2,1)
420 PRINT : :TAB(4);"SCORE =";INT(H*1
    00/T+.5);"PERCENT": :TAB(4);"TRY
    AGAIN? (Y/N)"
430 CALL KEY(0,KEY,S)
440 IF KEY=89 THEN 150
450 IF KEY=78 THEN STOP ELSE 430
460 CALL CLEAR :: CALL MAGNIFY(4)
470 CALL CHAR(96,"3C7EFFFFFFFF7E3C000
    00000000000000000000000000000000
    00000000000"):: CALL COLOR(2,7,16)
480 CALL HCHAR(9,6,42,21):: CALL HCHA
    R(13,6,42,21)
490 CALL VCHAR(10,6,42,3):: CALL VCHA
    R(10,26,42,3)
500 DISPLAY AT(11,6)SIZE(17):"T R A P
     S H O O T"
510 CALL CHAR(64,"3C4299A1A199423C")
530 CALL CHAR(104,"10381000000000000
    000000000000000000000000000000000"
    ):: CALL COLOR(2,16,7)
540 CALL CHAR(108,"18181C0E0607030101
    00000000000000000000000008080C0E
    060703818 18"):: CALL COLOR(2,7,16)
550 CALL CHAR(112,"000C0C0E0607030301
    010000000000000000000000008080C
    0C0E060703030"):: CALL COLOR(2,16,7)
560 CALL CHAR(116,"060606070303030101
    01000000000000000000000008080800C
    0C0C0E0606060"):: CALL COLOR(2,7,16)
570 CALL CHAR(120,RPT$("01",16)&RPT$(
    "80",16)):: CALL COLOR(2,16,7)
580 CALL CHAR(124,"000000000000010101
    03030307060606606060E0C0C0C080808
    "):: CALL COLOR(2,7,16)
590 CALL CHAR(128,"000000000000000101
    030307060E0C0C0030307060E0C0C0808
    "):: CALL COLOR(2,16,7)
600 CALL CHAR(132,"000000000000010103
    07060E1C181800181838706 0E0C08080"
    ):: CALL COLOR(2,7,16)
610 CALL COLOR(9,11,1):: CALL COLOR(2
    ,16,7)
620 CALL CHAR(140,"0103070F1F3F7FFFFF
    FFFFFFFFFFFFFF80C0E0F0F8FCFEFF0")
    :: CALL COLOR(2,7,16)
630 CALL CHAR(100,"DCDD590244D29B1900
    00000000000000000000000000000000
    "):: CALL COLOR(2,16,7)
640 CALL COLOR(14,11,1):: CALL COLOR(
    2,7,16)
650 CALL CHAR(136,"0"):: CALL COLOR(2
    ,16,7)
660 CALL CHAR(92,"82A6B7F7FFFFFFFF"):
    : CALL COLOR(2,7,16)
670 CALL COLOR(2,2,1):: CALL CLEAR
680 DISPLAY AT(3,1):"A CLAY PIGEON WI
    LL SPRING{5 SPACES}":"FROM THE TR
    AP."
690 DISPLAY AT(8,1):"USE THE ARROW KE
    YS TO AIM{4 SPACES}":"YOUR RIFLE
    LEFT OR RIGHT."
700 DISPLAY AT(13,1):"PRESS <ENTER> T
    O SHOOT."
710 DISPLAY AT(17,1):"YOU WILL HAVE A
     CHANCE{7 SPACES}":"TO SHOOT 50 C
    LAY BIRDS."
720 DISPLAY AT(24,1):"PRESS ANY KEY T
    O START."
730 CALL KEY(0,KEY,S):: IF S=1 THEN 1
    50 ELSE 730
740 END
```

©

Musical Scales On The VIC

Brian H. Lawler

"Scales" is a short, 2K RAM educational program which exploits the sound generating capabilities of the VIC-20 microcomputer. The program allows you to choose one of nine musical scales in the key of your choice. The computer then plays the scale up and down and assigns eight notes of the scale, in ascending order, to keys 1 through 8 on the VIC keyboard. You may then play any note on the scale by pressing one of these keys.

You will soon be able to play simple tunes on the scales "by ear," even if you can't read a note of music. Besides being fun, this exercise will give you some understanding of the scales used in different types of music. You will be able to recognize which scale is commonly used in jazz and which scale has an oriental sound. Get together with three of your computer friends and start a VIC quartet, or be the first composer on your block to write a symphony for cello and VIC.

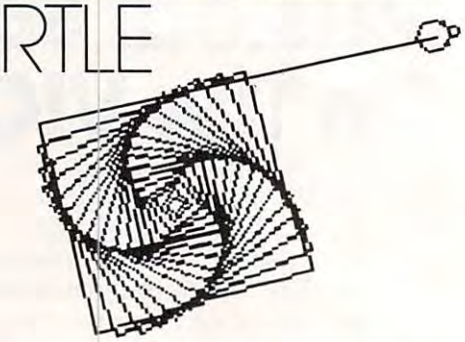
Program Notes

Line Nos.	
10	Initializes variables S1-S4 as the four VIC "voice" locations and variable VO as volume.
20	Reads data into variable N(x). These are all the notes that the VIC can play. The values are from page 135 of the *User's Guide.*
30-38	These are strings containing the data used in making the scales. A 1 raises the next note 1/2 step, a 2 raises the next note a whole step, etc.
155-160	Get the scale number.
175-240	Input the key and set variable S as the pointer to the first note of the scale.
250-290	Put the notes of the scale into Q(1) to Q(8) by using the data strings in lines 30-38.
300-390	Play the selected scale up and down once.
420-450	Get your note and POKE it into S2.
460	Waits for you to release the key.
470	Turns off the sound and goes back to line 420 to wait for another note.

```
5 DIMN(37)
10 VO=36878:S1=36874:S2=S1+1:S3=S2+1:S4=S
   3+1
20 FORI=1TO37:READN(I):NEXT
30 D$(1)="2212221"
31 D$(2)="2122122"
32 D$(3)="2122131"
33 D$(4)="2322323"
34 D$(5)="2222222"
35 D$(6)="2122212"
36 D$(7)="1222122"
37 D$(8)="2221221"
38 D$(9)="2212212"
100 PRINT"{CLEAR}          SCALES"
110 PRINT"{02 DOWN}"
112 PRINT"THIS PROGRAM ALLOWS   YOU TO SEL
    ECT A MUSI-";
114 PRINT"AL SCALE IN ANY KEY."
115 PRINT"THE COMPUTER THEN AS- SIGNS THE ~
    NOTE VALUES"
116 PRINT"TO KEYS 1 TO 8 ON THE VIC KEYBOA
    RD."
117 PRINT"{03 DOWN}HIT ANY KEY-"
118 GOSUB890
120 PRINT"{CLEAR}{02 DOWN}{REV}1{OFF} MAJOR"
122 PRINT"{DOWN}{REV}2{OFF} MINOR"
124 PRINT"{DOWN}{REV}3{OFF} HARMONIC MINOR"
126 PRINT"{DOWN}{REV}4{OFF} PENTATONIC"
128 PRINT"{DOWN}{REV}5{OFF} WHOLE TONE"
130 PRINT"{DOWN}{REV}6{OFF} DORIAN"
132 PRINT"{DOWN}{REV}7{OFF} PHRYGIAN"
134 PRINT"{DOWN}{REV}8{OFF} LYDIAN"
136 PRINT"{DOWN}{REV}9{OFF} MIXOLYDIAN"
150 PRINT"{02 DOWN}WHICH SCALE?"
155 GOSUB890:IFA$<"1"ORA$>"9"THEN155
160 SC=VAL(A$)
170 PRINT"{CLEAR}WHAT KEY?"
171 PRINT"{DOWN}         {REV}ABCDEFG{OFF}"
172 PRINT"{DOWN}    {REV}#{OFF} SHARP   {
    REV}-{OFF} FLAT"
173 PRINT
175 INPUT KY$
180 K$=LEFT$(KY$,1)
190 IFK$<"A"ORK$>"G"THENGOTO170
200 IFK$="C"THENS=13
202 IFK$="D"THENS=15
204 IFK$="E"THENS=17
206 IFK$="F"THENS=18
208 IFK$="G"THENS=8
210 IFK$="A"THENS=10
212 IFK$="B"THENS=12
220 IFLEN(KY$)=1THEN250
225 K$=RIGHT$(KY$,1)
230 IFK$="#"THENS=S+1:GOTO250
235 IFK$="-"THENS=S-1:GOTO250
240 GOTO170
250 Q(1)=N(S)
260 FORI=2TO8
270 S=S+VAL(MID$(D$(SC),I-1,1))
280 Q(I)=N(S)
290 NEXTI
300 REM-PLAY IT
305 POKEVO,15
310 FORI=1TO8
320 POKES2,Q(I)
330 FORK=1TO100:NEXT
340 NEXTI
350 FORI=7TO1STEP-1
360 POKES2,Q(I)
370 FORK=1TO100:NEXT
380 NEXTI
390 POKES2,0:POKEVO,0
400 PRINT"{CLEAR}YOU MAY NOW PLAY THE  SCA
    LE ON YOUR KEY-    BOARD."
410 PRINT"{02 DOWN}  --HIT {REV}↑{OFF} TO ~
    QUIT--"
420 GOSUB890:IFA$="↑"THENPOKES2,0:POKEVO,0
    :GOTO120
430 IFA$<"1"ORA$>"8"THEN420
440 A=VAL(A$)
450 POKEVO,15:POKES2,Q(A)
460 IFPEEK(203)<>64THEN460
470 POKES2,0:GOTO420
890 A$="":GETA$:IFA$=""THEN890
895 RETURN
900 DATA 135,143,147,151,159,163,167,175,1
    79,183,187,191
910 DATA 195,199,201,203,207,209,212,215,2
    17,219,221,223
920 DATA 225,227,228,229,231,232,233,235,2
    36,237,238,239,240
```

©

FRIENDS OF THE TURTLE



David D. Thornburg, Associate Editor

On Logo And Turtles

Last December I attended the California Math Council meeting at the Asilomar conference center. Although the conference was directed primarily towards educators from Northern California, attendees came from all over the country.

One evening I was giving an informal presentation in a hospitality suite. While the computer was running a graphics demonstration, one teacher came up to me and said, "I didn't know that Logo ran on the Atari 800."

"That's not Logo," I replied. "That's Atari PILOT."

"Oh," said the teacher, "I thought these pictures were made with turtle graphics."

"They *were* made with turtle graphics," I said. "Turtle graphics doesn't have any specific relationship to any one computer language."

As we talked some more, it became clear that Logo is becoming more and more identified with the turtle. Worse, the turtle is becoming more and more identified with Logo. While the teacher who approached me seemed startled to find that there were turtles outside of Logo, and that Logo could do far more than turtle graphics, I'm sure that this confusion is commonplace among new computer users.

The real tragedy comes when the association of Logo as simply a turtle graphics language becomes a self-fulfilling prophecy. I am content to believe that many Logo users may not want to use the other features of Logo for the first few months. But any language that has *just* turtle graphics, no matter how sophisticated, user-friendly, or Logo-like it is, is not Logo.

Radio Shack Color Logo

Unfortunately, the superb turtle graphics package developed for the Radio Shack Color Computer (Radio Shack Color Logo) is one example of such a language. Radio Shack Color Logo supports much of what we expect from Logo – extensibility, local variables, recursion, and turtle graphics. However, the only variables that can be used with this language are numbers. There is none of the list processing capability that gives Logo its tremendous power as a symbol manipulation language.

The sad part is that this symbol manipulation capability is often of value in advanced turtle graphics programs! If you doubt this, you can see some striking examples in Abelson and diSessa's *Turtle Geometry*, or in my new Logo book.

Even with these detractions, I find Radio Shack Color Logo to be a tremendous turtle graphics language. It supports multiple turtles (created with the word HATCH). It has a built-in procedure editor that allows the user to format multi-line statements so they look nice. (Most other Logos require that you just keep typing a line until you are done. For a line of a few hundred characters in length, this can look messy.)

Radio Shack Color Logo runs in a 32K system (using the disk-based version I had last year), and a cartridge version (as of this writing) is expected to run in a 16K computer. This makes Radio Shack Color Logo one of the less expensive Logo-like turtle graphics packages on the market.

Realtime Animation

The language provides the user with several modes. When the computer is turned on, it is in the BREAK mode. To gain access to the turtle immediately, you simply press R to enter the RUN mode. From this mode the turtle can draw pictures using single commands such as FD 50, RT 37, etc. Unfortunately, you cannot enter repeated commands in this mode. For example, you cannot enter REPEAT 4 (FD 30 RT 90) to draw a square. You must use this command inside a procedure instead.

A very nice feature of the RUN mode that has great appeal to young turtle users is the DOODLE mode. To enter this mode from the RUN mode, the user just types the character @. The computer then waits for the user to enter a word that becomes a procedure name. Once this has been entered, the user can draw pictures by pressing the number keys on the keyboard. Each key corresponds to a different command, e.g., CLEAR, HOME, PENUP, PENDOWN, RIGHT 45, LEFT 45, FORWARD 1, FORWARD 10, RIGHT 15, and LEFT 15. Once a picture is completed, the user can redraw it by simply entering the procedure name from the RUN mode.

The EDIT mode allows the user to create his or her own extensions to the language. These can be saved on disk or tape (for the disk-based version), or on tape (for the cartridge version).

This language also supports user-defined turtle shapes and multiple turtles. Unlike TI's Logo, the user-defined multiple turtles can each draw lines and actually rotate as their orientation is changed. The high speed of these turtles makes this language useful for some realtime animation applications.

Overall, I am quite impressed by this language. As a turtle graphics environment, it should be of great use to all owners of the Radio Shack Color Computer. My only criticism is that Radio Shack is calling the language Logo, when it is not Logo at all.

The task of educating the public *and* the manufacturers is an important one. After all, you wouldn't think you had purchased a car if it didn't have an engine in it. To call a language Logo, one must be able to perform list processing. It would have been much better if Radio Shack had called the language TurtleTalk, or some other catchy name.

But, until the customers come to understand that Logo is far, far more than just a turtle language, we can't be overly critical when a manufacturer makes the same mistake. ©

Direct Atari Disk Access

Andrew Lieberman

Here are three programs that make disk access easier, display the contents of any sector on disk, and allow you to save screen displays to disk. Caution: these techniques write directly to disk. Be certain that you fully understand how to use these methods or you risk damaging existing disk files. The program opens the door to many interesting and valuable applications. And it's only 67 bytes long.

Even with a fast-formatted disk, the Atari disk drive is slow for many applications if BASIC commands like INPUT, PRINT, PUT, and GET are used. With the machine language subroutine in this article, you can transfer the contents of a specified area of memory to disk, and vice versa, quickly and easily, eliminating the need for the slower BASIC commands.

Program 1 is the source code for the program. Type in the program on your Assembler/Editor, assemble it, and save it with "SAVE#D:SECRAM.OBJ‹601,643". If you do not have the Assembler/Editor, use Program 2. Type in the program, save it, run it, go to DOS, and use option K, binary save, by typing: D:SECRAM.OBJ‹601,643". It may be a good idea to lock the file.

To use this subroutine in a BASIC program, just add Program 2 to the BASIC program. Be sure the DATA has been put into memory before the routine is used; otherwise, you will crash the system. To call the routine, simply type: I=USR (1537,RAM,SECTOR,NUMSEC,DCOMD).

"I" can be any variable; RAM is the starting memory location; SECTOR is the first sector to be read from or written to. Each disk has 720 sectors, numbered from 1 to 720. The computer fills these sectors starting with 1 and works up, so you should plan to use sectors from 650 to 720 depending upon how many you need.

These sectors are not protected; if the disk starts getting full, your information may be overwritten. Program 3, which is described later, will be a help in preserving your data. NUMSEC is the number of sectors to be copied. There are 128 bytes to a sector and eight sectors to a kilobyte. DCOMD refers to read or write. An 82 here means read from disk to RAM, and 87 means copy memory to disk.

Let's look at an example of all this. Suppose you wanted to copy a modified character set to disk. Suppose further that your character set is located in memory locations 30720 through 31743 and you wanted it stored starting at sector 700. You would first have to calculate that you need eight sectors for the 1024 bytes of character set. Then simply type I=USR(1537,30720,700,8,87). If you did not understand this example, go back and look at what each number means; it should then be clear.

Saving Data And Graphics Displays

There are many applications for this program. Program 3 will display the contents of any sector on a disk. Another application that you are sure to find useful is saving screen displays to disk for quick recovery from within a program. Suppose you wanted to save your current Graphics 0 screen to disk. Simply type: I=USR(1537,PEEK(88)+PEEK(89)*256,680,8,87). Clear the screen and then type: I=USR(1537,PEEK(88)+PEEK(89)*256,680,8,82).

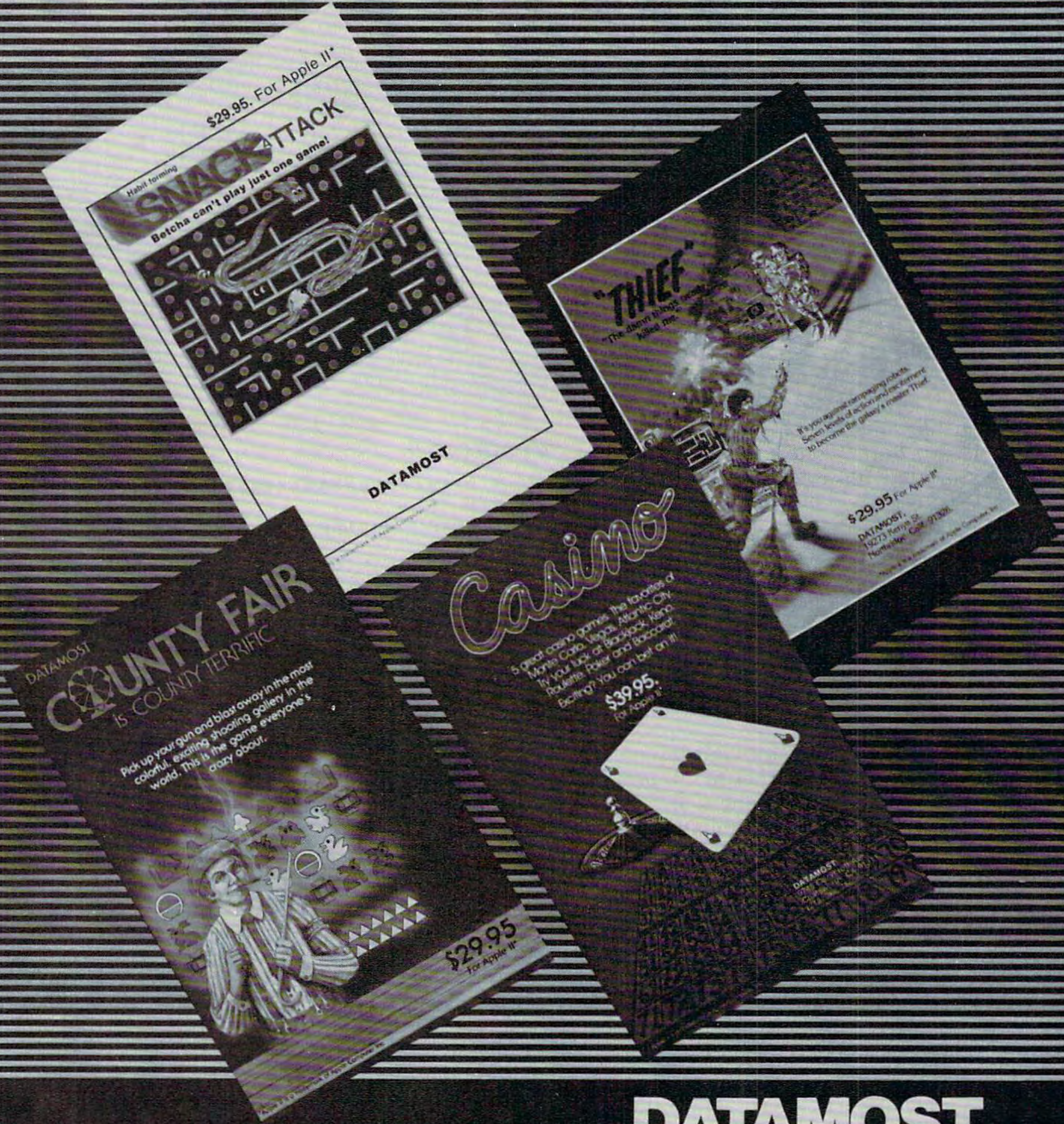
Voilà! After eight beeps you recover your old screen. If you have a customized display list, you may want to save it also by using: PEEK(560)+PEEK(561)*256 instead of PEEK(88)+PEEK(89)*256.

You should also find that this program works well when saving and loading character sets and player/missile data. The program should be used in any situation in which the contents of any area of memory should be the same every time, like a character set or a graphics display.

Program 3 is a simple program that copies the contents of a disk sector into a string and then prints the string on the screen. RETURNs are printed as "(RET)", and other editing characters are printed as their graphics symbols, i.e., with an ESCape printed first. This is very useful for finding free space on a disk for saving DATA. If, for example, you wanted to check sectors 700 to 710 to make sure they are empty, just RUN the program, start with 700, then use the right arrow to see what is on 701, etc.

A whole string of hearts (CHR$(0)) indicates an empty sector. Anything else means there is DATA on that sector. This program may also be used to modify DOS and other programs that

FUNTASTIC!


$29.95. For Apple II*
Habit forming
SNACK ATTACK
Betcha can't play just one game!
DATAMOST
"THIEF"
"The damn things killed me."
It's you against rampaging robots.
Seven levels of action and excitement
to become the galaxy's master Thief.
$29.95 For Apple II*
DATAMOST,
19273 Kenya St.
Northridge, Calif. 91326
DATAMOST
COUNTY FAIR
is COUNTY TERRIFIC
Pick up your gun and blast away in the most colorful, exciting shooting gallery in the world. This is the game everyone's crazy about.
$29.95
For Apple II*
Casino
5 great casino games, the favorites of Monte Carlo, Vegas, Atlantic City.
Try your luck at Blackjack, Keno
Roulette, Poker and Baccarat!
Exciting? You can bet on it!
$39.95.
For Apple II*
DATAMOST

DATAMOST

9748 Cozycroft Ave.,
Chatsworth, CA 91311
(213) 709-1202

cannot normally be modified. Look at sector 43 of any DOS II disk. It should be the top of the menu. If it isn't, find the correct sector. Now, BREAK the program and type "PRINT A$". It will be the same as what appeared on the screen except RETURNs and CLEAR SCREENs will be printed.

Try making some changes in the middle of the string. For example, type: A$(71,87) = "A. DISK MENU   ". Then save this modified string *back to the disk by typing: I* = USR(1537,ADR(A$), 43,1,87). Now go to DOS, and if all went correctly your change has been made.

Now that you know how to use this program, you probably want to know how it works. Lines 1 through 40 should be fairly obvious. Line 50 clears the keyboard of any key pressed earlier. Line 60 reads the keyboard. A 7 means the right arrow was hit, so the variable SEC is incremented. Line 70 checks for a left arrow in the same way. If no key has been pressed, the program jumps back to line 60 to wait for a key to be pressed. If a key other than left or right arrow was pressed, line 90 accepts the input.

Lines 100 and 110 check to make sure the sector is within the legal limit. Line 130 loads the requested sector into the RAM area of string A$. Instead of just printing the string to the screen, each character is printed one at a time. Before the character is printed, it is checked for being a RETURN (CHR$(155)); if it is a RETURN, "(RET)" is printed instead. Furthermore, an ESCape is printed before each character. If these precautions were not taken, many sectors would clear the screen and do other strange, undesirable things when printed. The extra spaces are printed at the end of the sector to clear away any loose ends left over from the last sector.

Easy Programming

Now for the good stuff: how does this program work in only 67 bytes? The real key to this program is the Operating System subroutine at $E453. Each time it is JSRed to, it takes the information in the lower page three memory locations and processes it, and it does that very quickly. There are many handy subroutines in the Operating System for things like print to the screen, plot, drawto, set up VBLANK, change graphics modes, etc. For more information on how to use the graphics subroutines, get the February 1982 issue of **COMPUTE!** and look at "Insight: Atari," page 77. These subroutines can make life very easy on a programmer.

You should be able to interpret how the assembly language program works by looking at the comments in the source code. The only part that is likely to be unfamiliar to you is the first part. The first number in the USR command is the starting memory location. The other numbers are all placed on the stack as shown in the table. Lines 260 to 390 pull the values off the stack and put them into the memory locations in which they belong.

There is one other memory location that you may find useful: $303 (decimal 771) shows the status after the most recent operation. A 1 means everything is all right. Any other number is an error code. Errors are usually the result of trying to read a bad, or nonexistent, sector.

Top Of Stack

Number of variables passed		xx	We ignore this
First number passed	hi byte	xx	RAM pointer
First number passed	lo byte	yy	RAM pointer
Second number passed	hi byte	xx	Sector pointer
Second number passed	lo byte	yy	Sector pointer
Third number passed	hi byte	xx	Number of sectors – we ignore
Third number passed	lo byte	yy	the hi byte because the program is set up to do a maximum of 255 sectors
Fourth number passed	hi byte	xx	Disk command – since this
Fourth number passed	lo byte	yy	value should only be $52 or $57 we can ignore the hi byte

Bottom Of Stack

Program 1.

```
0100 ;*********************************
    **
0110 ;**A routine for storing RAM on
    **
0120 ;**a disk or for reading it back
    **
0130 ;**by ANDREW LIEBERMAN  7/10/82
    **
0140 ;*********************************
    **
0150 NUMSEC=$600 ;Number of sectors s
    till to be done
0160 DUNIT=$301 ;Which drive?(1-4)
0170 DCOMD=$302 ;$52=Read, $57=Write
0180 DBUFLO=$304 ;Pointer for Lo byte
     of RAM
0190 DBUFHI=$305 ;Pointer for Hi byte
     of RAM
0200 DAUXLO=$30A ;Pointer for Lo byte
     of sector
0210 DAUXHI=$30B ;Pointer for Hi byte
     of sector
0220  *=$601
0230 ;The USR command places data on
    the stack
0240 ;This part of the program pulls
    the data off and puts it in the
0250 ;proper memory locations
0260  PLA ;We don't care about this
0270  PLA
0280  STA DBUFHI
0290  PLA
0300  STA DBUFLO
0310  PLA
0320  STA DAUXHI
0330  PLA
0340  STA DAUXLO
0350  PLA ;This is assumed to be 0
0360  PLA
```

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

ATARI is a registered trademark of ATARI, Inc.

```
0370   STA NUMSEC
0380   PLA ;This is assumed to be 0
0390   PLA
0400   STA DCOMD
0410   LDA #$01 ;Assume drive 1
0420   STA DUNIT
0430 LOOP DEC NUMSEC ;One less sector
     to be done
0440   BMI END  ;If minus result, last
     sector was 0, so branch to END
0450   JSR $E453 ;This is the O.S. sub
    routine that does all the work
0460   CLC
0470   INC DAUXLO ;Increment sector po
    inter
0480   BCC SKIP1 ;Check for carry
0490   INC DAUXHI ;There was a carry s
    o hi byte is incremented
0500 SKIP1 LDA DBUFLO ;Since each sec
    tor is $80 bytes long, the
0510   CLC ;RAM pointer, DBUF, must be
     incremented by $80
0520   ADC #$80 ;Add $80 to lo byte
0530   BVC SKIP2 ;If it didn't overflo
    w everything's O.K.
0540   INC DBUFHI ;Lo byte overflowed,
     so increment hi byte
0550 SKIP2 STA DBUFLO ;Don't forget t
    o store the lo byte
0560   CLC ;A jump done this way makes
     the program relocatable in RAM
0570   BCC LOOP
0580 END RTS ;All done
```

Program 2.

```
1 GOSUB 31000
30999 END
31000 RESTORE 31010:FOR I=1537 TO 160
     3:READ J:POKE I,J:NEXT I:RETURN

31010 DATA 104,104,141,5,3,104,141,4,
     3,104,141,11,3,104,141,10,3,104
     ,104,141,0,6,104,104,141,2,3,16
     9,1,141,1,3,206
31020 DATA 0,6,48,29,32,83,228,24,238
     ,10,3,144,3,238,11,3,173,4,3,24
     ,105,128,80,3,238,5,3,141,4,3,2
     4,144,222,96
```

Program 3.

```
1 REM A program to examine disk secto
  rs
5 GOSUB 31000
10 DIM A$(128):A$(128)=" "
20 GRAPHICS 0:? "Type sector number,
   or use right arrowfor next sector,
    left arrow for last  sector."
30 POSITION 2,5:? "{11 M{CLEAR}"
40 POSITION 2,12:? "{4 DEL-LINE
   {CLEAR}WHAT SECTOR?";
50 POKE 764,255
60 I=PEEK(764):IF I=7 THEN SEC=SEC+1:
   GOTO 100
70 IF I=6 THEN SEC=SEC-1:GOTO 100
80 IF I=255 THEN 60
90 TRAP 40:INPUT SEC
100 IF SEC<1 THEN SEC=1
110 IF SEC>720 THEN SEC=720
120 POSITION 2,4:? "SECTOR #";SEC;"
   "
130 I=USR(1537,ADR(A$),SEC,1,82):POSI
   TION 2,6:FOR I=1 TO 128:J=ASC(A$(
   I,I))
140 IF J=155 THEN ? "(RET)";:GOTO 160
150 ? CHR$(27);CHR$(J);
160 NEXT I:? "{28 SPACES}":GOTO 40:REM
   about 30 spaces
30999 END
31000 RESTORE 31010:FOR I=1537 TO 160
     3:READ J:POKE I,J:NEXT I:RETURN

31010 DATA 104,104,141,5,3,104,141,4,
     3,104,141,11,3,104,141,10,3,104
     ,104,141,0,6,104,104,141,2,3,16
     9,1,141,1,3,206
31020 DATA 0,6,48,29,32,83,228,24,238
     ,10,3,144,3,238,11,3,173,4,3,24
     ,105,128,80,3,238,5,3,141,4,3,2
     4,144,222,96
```

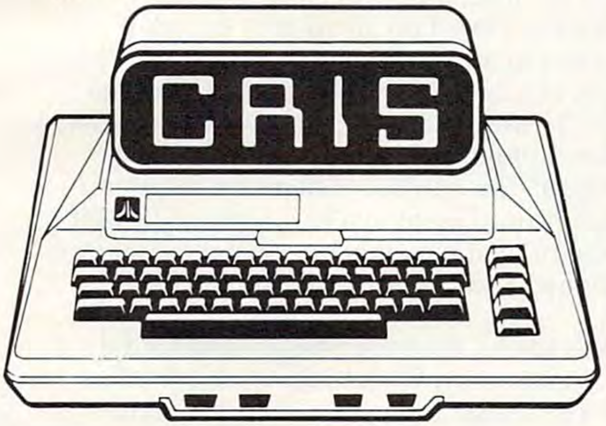
©

Do You Run A Retail Business?

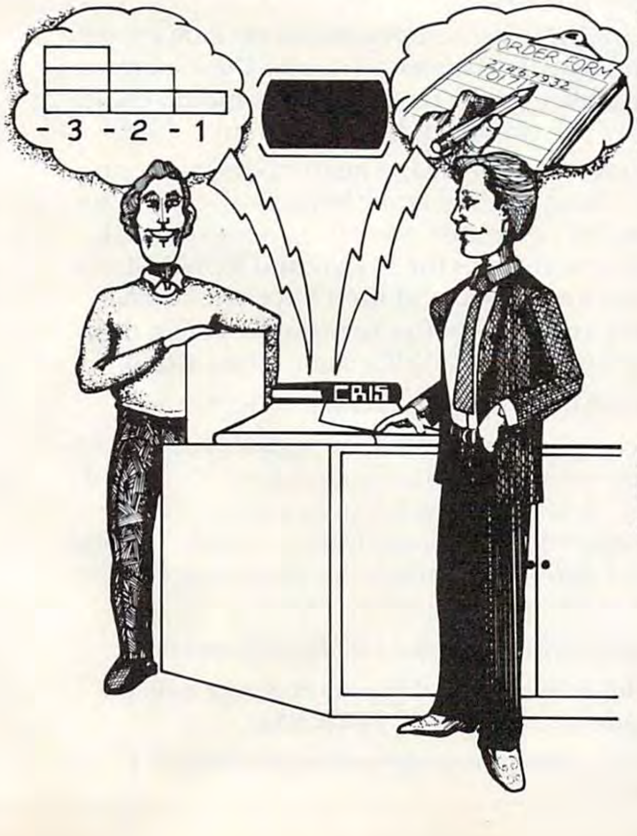
The Computer Age Comes To Your Retail Business Via C.R.I.S.

The computer age comes to your retail business via C.R.I.S. (Cash Register Inventory System) — a complete software package that will literally convert your ATARI 400 or 800 into a total point-of-sales terminal. C.R.I.S. is a complete inventory control system, too! Information on backorders, total items sold, items in stock, stock ordered, vendor numbers — it's all here and at your fingertips. C.R.I.S. also prints complete inventory reports, including purchase order, stock inventory, end of period, AND it will print customer sales receipts, just like the "Big Machines" do! at 1/10th the price.

THE PROGRAM for Store Managers



Retailers — Just think of it: a complete inventory and point-of-sales system in a single package!



- ★ Holds up to 1000 separate inventory items (1400 with C.R.I.S. II)
- ★ Supports up to 10 salesmen
- ★ User-defined commission rate for each salesman
- ★ Accepts stock name of up to 20 characters in length
- ★ Up to 16 items can be listed on each invoice
- ★ End of Day, End of Period, and Purchase Order reports
- ★ Easy error correction and recovery
- ★ Many "password-protected" functions to help prevent accidental data loss
- ★ Calculates totals on sales tickets and computes change
- ★ Requires only 1 disk drive (2 required for C.R.I.S. II)

PACKAGE INCLUDES C.R.I.S. II

This latest version features the dependability of C.R.I.S. I with the added punch of a 40% increase in storage capacity. Since C.R.I.S. II requires an additional 8K of memory (for a total of 40K) and two disk drives, you can purchase C.R.I.S. I for use now on your one-drive, 32K system, and move up to C.R.I.S. II whenever you decide to expand your system.

C.R.I.S. has been designed for use with the following equipment:

- ★ ATARI 400 or 800 with 32K memory (40K required for C.R.I.S. II)
- ★ Atari 810 disk drive (2 drives required for C.R.I.S. II)
- ★ Black & white or color TV
- ★ 40-column printer (if customer receipts are to be issued)
- ★ 132-column printer (for report information)

We recommend having both printers (receipts can be run on the 132-column printer) and a second disk drive (to make backups easier). These peripherals are optional and can be added later.

$249⁹⁵

THE BUSINESS DIVISION
A Subsidiary of Scott Adams, Inc

THE BUSINESS DIVISION
P.O. BOX 3435
LONGWOOD, FL 32750
(305) 862-6917

Automatic Commodore Program Selector

Steven A Smith

Here are several ways to make disks easier to use with the VIC, Commodore 64, or PET/CBM. Are you setting up a business application and want to save users the trouble of working with LOAD and RUN? Are you planning a party and want to avoid spending all your time just showing friends how to run the games? Do you want to save yourself time and trouble in your everyday computing? Try out these various menu programs and see if they wouldn't be useful in many ways.

If you want to be able to choose from among a number of options within a program, one of the best methods available is a menu. The computer displays a list of items with numbers or letters assigned to each, and you press the number or letter corresponding to the option you want. This way, you don't have to worry about which responses are allowed or about how to spell a particular response, and it's much faster.

All this applies to disk drives, as well. Also, someone who is not familiar with the operating system of the computer can call up any of a number of programs without having to know about diskette directories or about LOADing or RUNning programs.

You can choose between two ways of automating program selection from a disk. The first one we'll describe uses specific, pre-defined menus for each diskette or function. The second can be used with any diskettes, determining at runtime which programs are available on the disk.

Pre-defined Menus

A pre-defined menu is written right into the BASIC menu program. Because of this, a new program must be written for each diskette for which you want a menu. However, there are several advantages to using a pre-defined menu. First, it's fast. As soon as you RUN it, the menu program knows what programs should be on the diskette and can go about the business of displaying the menu. Also, you can add program descriptions to the menu screens to show more information about the programs than just their names.

Another, less obvious advantage to pre-defined menus is that you can set up a menu for just a few of the programs on a diskette, have another menu for some others, and have other programs that are not accessed by any menus. This way, you can let someone have access to only the programs that a particular application requires.

Program 1 is a sample of a pre-defined menu for an inventory file maintenance system. Although it is short, it is surprising how impressive it can be in operation, especially to someone who is used to having to load and run individual programs via the traditional directory method.

Lines 120-130 set up an array of program names, one per array element.

Lines 140-230 display the actual menu. The numbers "1" through "8" are displayed in reverse, with a description of the associated programs next to them. The number of items on the menu is not significant – eight just happened to fit well on this menu. Just remember to change your array dimensioning and the error-checking in lines 250-260.

In this menu, the programs are grouped by type of operation to make things clearer for the user. Inventory file operations, transaction file operations, and setup operations are each grouped together and separated from the others by a line. Of course, you can display and group items on your menus any way you wish, remembering to have your item numbers and array elements correspond properly.

Lines 240-260 accept your menu item choice, making sure it is between one and the maximum item number on the menu. On this menu, choice number "8" simply ends the program.

Lines 270-300 are the heart of the menu program. Using the "dynamic keyboard" technique (where the computer *enters its own instructions*) the computer types the LOAD and RUN instructions on the screen, and then forces RETURNs into the keyboard buffer to make it execute them. For Original ROM BASIC, change line 300 to:

```
300 POKE 527,13:POKE 528,13:POKE 525,2
```

To accomplish this "dynamic" effect, you need to POKE a value of 13 into the first two "Keyboard Buffer" bytes, and a value of two into "Number of Characters in Keyboard Buffer." These locations vary on different Commodore machines. For the VIC and the 64, change line 300 to:

```
300 POKE 631,13:POKE 632,13:POKE 198,2:END
```

Line 300 in the printed program works as is on Upgrade and 4.0 BASIC PET/CBMs.

This menu program will expect to find a "Li-

WORDCRAFT 20 ...A First! Word Processing for VIC 20®

Never be at a loss for words again! Expressing yourself is easier, faster, and less expensive than ever before . . . thanks to Wordcraft 20© from United Microware. Wordcraft 20© has all the features found on bigger systems costing a lot more — all the features you'll need to prepare "picture-perfect" documents, letters, personalized mailers, lists, and many other projects that have you juggling words.

Wordcraft 20© is easy to use; it won't confuse you. Your words will appear on the screen just as you've written them. This fully featured system lets you do what you need to do . . . change a character, a word, or an entire block of copy . . . arrange and rearrange the text . . . set the tabs and margins . . . adjust the line lengths . . . move the copy up and down, left or right, so you see every word on the screen . . . before you print. Wordcraft 20© gives you personal word processing with professional results. Plug Wordcraft 20© into your VIC® today . . . see for yourself just what words will do. Available exclusively from your local UMI dealer.

umi

United Microware Industries, Inc.
3503-C Temple Avenue
Pomona, CA 91768
(714) 594-1351

Commodore and VIC 20 are registered trademarks of Commodore Business Machines, Inc. Wordcraft 20© copyright held by P.L. Dowson.

brary Inventory System" diskette in Drive 1. If you want to use Drive 0, just change the "1:" in line 280 to "0:".

Increasing Menu Items

Nine items can be placed on this menu before it begins to look crowded. There are two ways to improve on this number: the first is simply to use several menus and let each menu chain (call in) the next. You can let one menu item be the next menu program, or add a line:

```
245 IF A$=CHR$(13) THEN C$(1)="MENU2":A$="
    1":GOTO 27Ø
```

This line will call the next menu program (here named MENU2) if RETURN, rather than one of the options shown, is pressed.

While this works quite well, you do have to wait for the new menu to be loaded each time you chain from one to the next. A faster way is shown in Program 2. Several menus can be stored in the same program. By pressing RETURN, you can go from one menu to the next without waiting to load a new menu program. A message is added to the bottom of the screen indicating that you can press RETURN to go on to the next menu. After the last menu is shown, pressing RETURN again will bring you back to the first menu. Of course, going to the next menu could itself be made a menu option, instead of being automatic.

To make menus especially useful to people unfamiliar with computers, you can make the programs called by the menu, themselves call the menu back when they finish. To do this, find where your program ends, whether by an END statement or by reaching the last of the line numbers. Change your END statements to GOTO 62000 and add the following lines:

```
62ØØØ PRINT"{CLEAR}{Ø4 DOWN}"
62Ø1Ø PRINT"LOAD"CHR$(34)"Ø:MENU"CHR$(34)",
    8{Ø4 DOWN}"
62Ø2Ø PRINT"RUN":PRINT"{Ø9 UP}"
62Ø3Ø POKE 623,13:POKE 624,13:POKE 158,2:END
```

This assumes that your menu program is named "Menu" and is in Drive 0. As before, change line 62030 for your computer exactly as you modified line 300, to perform the "dynamic keyboard" on your model.

Once you load the menu program, you don't need to worry about loading any more programs. Each time you finish one program, the machine will take you back to your menu. This is why menus are especially helpful for inexperienced operators. A menu also works well at parties – you set it up with games which call back the menu, and you don't have to worry about being around to show people how to load and run their choices.

Fully Automatic Menus

Program 3 is a different method of generating menus, a fully automatic diskette menu. When you run this program, you can put any disk in Drives 0 and/or 1, and it will find out what programs are on the disk and build a menu around them. Although you can't add descriptions to the program names, with diskette files you do have 16-character names to work with, and you can make *them* quite descriptive.

This method is slower than using pre-defined menus because, before the program can generate the menus, it must read the diskette directory and fill its own array of program names. However, you don't have to write a new menu program for each diskette or change a menu program when you change the contents of a diskette.

The following is a description of the variables used in Program 3:

AE$	: Filename Array
AN	: Array Entry Number
A0	: Files From Drive 0
C$	: Character Read In
DE	: Directory Entry
DR$	: Drive Number
ER	: Disk Error Number
F$	: Filename Found
FL	: Filename Length
I	: Iteration Variable
J	: Iteration Maximum
MM	: Maximum # On Menu
MN	: Menu Number

Lines 190-210 set up the variables and the program name array used by the program. Line 220 initializes the diskette in the drive currently being checked. Although the 4040 and 8050 diskette drives do not need to be initialized, this sets things up for line 230, which checks to see if a diskette was found in the drive. If not, the program goes over to the other drive.

Lines 240-250 are in the program mostly to let you know something is happening. While the program is reading the diskette directory, it lets you know how many programs it has found on that drive.

In lines 260-390, the diskette directory is opened and read as a sequential file. After skipping over the directory header, each directory block of eight file entries is checked for programs until the last entry is reached.

Line 310 skips entries which have their first byte equal to anything other than 130. That would indicate that the file was not a program file. You could use this line to create menus which displayed only USR or SEQ files if you wished. Line 330 puts the program name into string F$. Line 340 keeps the "Universal Wedge" DOS Support program from showing up on the menus. This line can be deleted if you wish. Line 350 updates

CALC RESULT™

Finally . . . A More Powerful Planning And Forecasting Tool That Takes Less Time, Work And Money Than Any Other On The Market!

With all the knowledge and experience gathered through previously released spread-sheet programs, it had to be possible to create a tool that would be easier to understand, easier to handle, and more powerful than those which already existed. And thus a new product was born!

CALC RESULT is just that . . . the result of dynamic enhancements to earlier versions of electronic spread-sheet programs! Below are listed just some of the highlights of this revolutionary new product!

- A three-dimensional spread-sheet with a minimum of 32 pages of 63X254 cells
- Utilizes only the memory in cells that are active
- Consolidation of pages
- Graphics (Histograms) on screen and printer
- Flexible Print format can be different than screen format
- View as many as three spread-sheets at one time through a window and split screen
- Replicate, copy and move commands
- Help function on-line
- Cells containing formulas are protected
- IF-THEN-ELSE with AND, OR and NOT-ELSE gives you unlimited possibilities in each cell
- RANDOM function
- Independent column width for column "A"
- Ability to Edit Formulas within a Cell
- Can load VisiCalc™ files

Contact Your Nearest Commodore Dealer Today . . .
You'll Be So Glad You Did!

For: Commodore 64 - Cartridge/Tape Version - $69.95
Commodore 64 - Cartridge/Disk Version - $149.95
Commodore 8032/8096 with 8050 Disk Drive - $199.00

Distributed By:
COMPUTER MARKETING SERVICES INC.
300 W. Marlton Pike, Suite 26
Cherry Hill, New Jersey 08002
(609) 795-9480

A Product Of:
handic
software ab
Box 42054
Vretensborgsvagen 8
126 12 Stockholm, Sweden

VisiCalc is a trademark of VisiCorp
CALC RESULT is a trademark of Handic Software ab

your screen to tell you how many program entries have been found, and line 360 puts this program name and drive number into the array of filenames found. Lines 370-380 then read past the proper number of bytes to be ready to read in the next file entry.

Lines 410-420 finish up the work with one drive and switch to the other, if necessary. If no programs are found on either drive, the program ends here. Otherwise, the first menu is ready to be displayed.

Entering Your Choices

Line 440 prints the menu heading. The heading will include a menu number starting with "1" and going as high as necessary to show all of the program names found, in groups of nine. Line 450 checks to see if there are enough program names left in the array to display nine menu items. If not, the menu is shortened. Line 460 displays the menu item itself, and lines 470-480 display the messages at the bottom of the screen.

Lines 490-530 check for your choice of menu item. It must be between "1" and the maximum number on the menu, or it can be RETURN, in which case the program will display the next menu. If there are no more items in the program name array, the first menu is redisplayed.

If the key you pressed was one of the menu items shown, the program continues to line 540. Variable AE$ is now the drive number, a colon, and the 16-character name of the program you have chosen. Any blanks in the name are stored in the directory as *shifted* spaces, with an ASCII value of 160.

Lines 560-580 check to see how long the program name is by looking backwards from the end for the first character that is not a shifted space. When one is found, variable FL contains the length of the name plus the drive number. Then, the LOAD and RUN instructions are displayed, and the keyboard buffer is POKEd with RETURNs to load the chosen program, just as in the pre-defined menu programs. Line 600 of Program 3 should be modified as before for the "dynamic keyboard" appropriate to your model.

Using these programs with the 2020 disk drive requires no changes. Using them with the 8050 drive requires only one change in Program 3. Change line 230 to:

```
230 IF DS=21 THEN 400
```

Program 1.

```
100 REM ** LIBRARY INVENTORY SYSTEM DRIVER
     MENU **
110 POKE59468,14:DIMC$(6)
120 C$(0)="SLIB":C$(1)="SLIBPRINT":C$(2)="
    SLIBINQ":C$(3)="STRANPRINT"
130 C$(4)="STRANPURGE":C$(5)="SLIBSETUP":C
    $(6)="FORMAT"
140 PRINT"{CLEAR}{02 DOWN}          {REV} ~
    PROGRAM CHOICE MENU {OFF}{02 DOWN
    DOWN}"
150 PRINT"        {REV}1{OFF} INVENTORY FIL
    E MAINTENANCE{DOWN}"
160 PRINT"        {REV}2{OFF} INVENTORY FIL
    E LISTING{DOWN}"
170 PRINT"        {REV}3{OFF} INVENTORY FIL
    E INQUIRY{02 DOWN}"
180 PRINT"        {REV}4{OFF} TRANSACTION F
    ILE LISTING{DOWN}"
190 PRINT"        {REV}5{OFF} TRANSACTION F
    ILE PURGE{02 DOWN}"
200 PRINT"        {REV}6{OFF} FIRST-TIME FI
    LE SETUP{DOWN}"
210 PRINT"        {REV}7{OFF} FORMAT A DISK
    ETTE{02 DOWN}"
220 PRINT"        {REV}8{OFF} END OF LIBRAR
    Y WORK{DOWN}"
230 PRINT"         {REV} CHOOSE ONE OF THE ~
    ABOVE {OFF}";
240 GETA$:IFA$=""THEN240
250 IFA$<"1"ORA$>"8"THEN240
260 IFA$="8"THENEND
270 PRINT"{CLEAR}{06 DOWN}"
280 PRINT"LOAD"CHR$(34)"1:"C$(VAL(A$)-1)CH
    R$(34)",8"
290 PRINT"{04 DOWN}RUN":PRINT"{09 UP}"
300 POKE623,13:POKE624,13:POKE158,2:END
```

Program 2.

```
100 REM ** INVENTORY SYSTEM DISKETTE DRIVE
    R MENU #1 **
110 POKE59468,14:DIMC$(9)
120 C$(1)="SLIB":C$(2)="SLIBPRINT":C$(3)="
    SLIBINQ":C$(4)="STRANPRINT"
130 C$(5)="STRANPURGE":C$(6)="SLIBSETUP":C
    $(7)="FORMAT":C$(8)="DIRECT"
140 PRINT"{CLEAR}{DOWN}          {REV} LIBRAR
    Y INVENTORY MENU 1 {OFF}{02 DOWN}
    "
150 PRINT"        {REV}1{OFF} LIBRARY FILE ~
    MAINTENANCE{DOWN}"
160 PRINT"        {REV}2{OFF} LIBRARY FILE ~
    LISTING{DOWN}"
170 PRINT"        {REV}3{OFF} LIBRARY FILE ~
    INQUIRY{02 DOWN}"
180 PRINT"        {REV}4{OFF} TRANSACTION F
    ILE LISTING{DOWN}"
190 PRINT"        {REV}5{OFF} TRANSACTION F
    ILE PURGE{02 DOWN}"
200 PRINT"        {REV}6{OFF} SETUP INVENTO
    RY FILES{DOWN}"
210 PRINT"        {REV}7{OFF} FORMAT A DISK
    ETTE{DOWN}"
220 PRINT"        {REV}8{OFF} PRINT A DISKE
    TTE DIRECTORY{02 DOWN}"
230 PRINT"      {REV}    CHOOSE ONE OF THE ~
    ABOVE    {OFF}"
240 PRINT"      {REV} OR PRESS RETURN FOR N
    EXT MENU {OFF}";
250 GETA$:IFA$=""THEN250
260 IFA$=CHR$(13)THEN290
270 IFA$<"1"ORA$>"8"THEN250
280 GOTO450
290 C$(1)="SLIBPRT1":C$(2)="SLIBPRT2":C$(3
    )="SLIBPRT3":C$(4)="SLIBPRT4"
300 C$(5)="SLIBPRT5":C$(6)="SLIBPRT6":C$(7
    )="SLIBPRT7":C$(8)="SLIBPRT8"
310 PRINT"{CLEAR}{DOWN}          {REV} LIBRAR
```

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

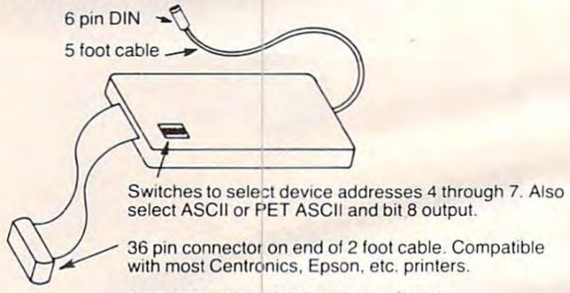
BOX 274 KINGSBRIDGE STN., RIVERDALE, N.Y. 10463 (212) 796-6200

```
    Y INVENTORY MENU 2 {OFF}{Ø2 DOWN}
    "
320 PRINT"          {REV}1{OFF} PRINT SALES R
    EPORT{DOWN}"
330 PRINT"          {REV}2{OFF} PRINT BACKORD
    ER REPORT{DOWN}"
340 PRINT"          {REV}3{OFF} PRINT DELINQU
    ENT ACCOUNTS{DOWN}"
350 PRINT"          {REV}4{OFF} PRINT HISTORI
    CAL REPORT{DOWN}"
360 PRINT"          {REV}5{OFF} PRINT HISTORI
    CAL SUMMARY{DOWN}"
370 PRINT"          {REV}6{OFF} PRINT SALES T
    AX REPORT{DOWN}"
380 PRINT"          {REV}7{OFF} PRINT MONTHLY
     REPORTS{DOWN}"
390 PRINT"          {REV}8{OFF} PRINT YEARLY ~
    REPORTS{DOWN}"
4ØØ PRINT"       {REV}    CHOOSE ONE OF THE ~
    ABOVE     {OFF}"
41Ø PRINT"       {REV} OR PRESS RETURN FOR N
    EXT MENU {OFF}";
42Ø GETA$:IFA$=""THEN42Ø
43Ø IFA$=CHR$(13)THEN12Ø
44Ø IFA$<"1"ORA$>"9"THEN42Ø
45Ø PRINT"{CLEAR}{Ø6 DOWN}"
46Ø PRINT"LOAD"CHR$(34)"Ø:"C$(VAL(A$))CHR$
    (34)",8"
47Ø PRINT"{Ø4 DOWN}RUN":PRINT"{Ø9 UP}"
48Ø POKE623,13:POKE624,13:POKE158,2:END
```

Program 3.

```
19Ø AE$="":AN=Ø:AØ=Ø:C$="":DE=Ø:DR$="Ø"
2ØØ ER=Ø:F$="":FL=Ø:I=Ø:J=Ø:MM=Ø:MN=Ø
21Ø POKE59468,14:DIMAE$(3ØØ)
22Ø OPEN15,8,15:PRINT#15,"I"+DR$
23Ø INPUT#15,ER:IFER=21THEN4ØØ
24Ø PRINT:PRINT"{DOWN}READING DIRECTORY OF
     DRIVE "DR$
25Ø PRINT"{DOWN}  PROGRAMS FOUND:  Ø"
26Ø OPEN8,8,8,"$"+DR$+",SEQ"
27Ø FORI=1TO254:GET#8,C$:NEXT
28Ø FORDE=1TO8:F$="":GET#8,C$
29Ø IFC$=CHR$(13)THEN41Ø
3ØØ IFC$=""THENJ=29:GOTO37Ø
31Ø IFASC(C$)<>13ØTHENJ=29:GOTO37Ø
32Ø AN=AN+1:J=11:GET#8,C$:GET#8,C$
33Ø FORI=1TO16:GET#8,C$:F$=F$+C$:NEXT
34Ø IFLEFT$(F$,9)="UNIVERSAL"ANDMID$(F$,11
    ,5)="WEDGE"THENAN=AN-1:GOTO37Ø
35Ø PRINT"{UP}"TAB(18)AN-AØ
36Ø AE$(AN)=DR$+":"+F$
37Ø FORI=1TOJ:GET#8,C$:NEXT
38Ø IFDE<>8THENGET#8,C$:GET#8,C$
39Ø NEXT:GOTO28Ø
4ØØ PRINT"{DOWN}NO DISKETTE FOUND IN DRIVE
     "DR$"{DOWN}":FORI=1TO2ØØØ:NEXT
41Ø CLOSE8:CLOSE15
42Ø IFDR$="Ø"THENDR$="1":F$="I1":AØ=AN:GOT
    O22Ø
43Ø IFAN=ØTHENPRINT"{Ø2 DOWN}{REV} NO PROG
    RAMS FOUND {OFF}{Ø2 DOWN}":END
44Ø MM=9:PRINT"{CLEAR}{DOWN}         {REV} ~
    PROGRAM CHOICE MENU #"STR$(MN+1)"
     {OFF}{DOWN}"
45Ø FORI=1TO9:IFAE$(MN*9+I)=""THENMM=I-1:I
    =9:GOTO47Ø
46Ø PRINTTAB(12)"{REV}"RIGHT$(STR$(I),1)"{
    OFF} "MID$(AE$(MN*9+I),3,16)"{DOWN}"
47Ø NEXT:PRINT"     {REV}    CHOOSE ONE OF ~
    THE ABOVE OR    {OFF}"
48Ø PRINT"      {REV} PRESS RETURN TO GO TO ~
    NEXT MENU {OFF}";
49Ø GETC$:IFC$=""THEN49Ø
5ØØ IFC$<>CHR$(13)THEN53Ø
51Ø MN=MN+1:IFMN*9+1>ANTHENMN=Ø
52Ø GOTO44Ø
53Ø IFC$<"1"ORVAL(C$)>MMTHEN49Ø
54Ø AE$=AE$(MN*9+VAL(C$))
55Ø PRINT:PRINT"{CLEAR}{Ø4 DOWN}MENU ITEM ~
    CHOSEN: #"C$" - "MID$(AE$,3,16)
56Ø FORI=18TO1STEP-1:FL=I
57Ø IFASC(MID$(AE$,I,1))<>16ØTHENI=1
58Ø NEXT:PRINT"{Ø4 DOWN}LOAD"CHR$(34)LEFT$
    (AE$,FL)CHR$(34)",8{Ø4 DOWN}"
59Ø PRINT"RUN":PRINT"{Ø9 UP}"
6ØØ POKE623,13:POKE624,13:POKE158,2:END
```

©


6 pin DIN
5 foot cable
Switches to select device addresses 4 through 7. Also select ASCII or PET ASCII and bit 8 output.
36 pin connector on end of 2 foot cable. Compatible with most Centronics, Epson, etc. printers.

Input Into Apple's EXEC

Wally Hubbard

This simulation of the INPUT command, written in Applesoft BASIC, can be used to make EXEC files take input from the Apple keyboard.

Normally, the command INPUT A$ in an EXEC file ignores the keyboard and uses the next line in the EXEC file as its input. As an example, the file

INPUT A$
INPUT B$
PRINT A$, B$

would set A$ = "INPUT B$".

Program 1 shows a text file, EXPUT, which issues a prompt and then puts the response from the keyboard into XX$. The second line then RUNs the file named by XX$. Program 2 shows an Applesoft BASIC program which can be used to make EXEC files. It could be used to enter Program 1, but because EXPUT is so long, Program 2 contains a subroutine that automatically enters EXPUT whenever you type CTRL-I.

Let me explain how EXPUT works. It uses two FOR/NEXT loops as WHILE-WENDs, which are not explicitly available in Applesoft. The FOR/NEXT loop using B keeps cycling until B = 1. B does not equal 1 until a key has been pressed. The statement B = (X>127) sets B = 1 if the statement in parentheses is true, otherwise B = 0. And X, the value at the keyboard port, is always less than 128 until a key is pressed. The B loop gets each character, and the A loop, which is around it, puts each character into XX$ until RETURN is pressed. The sequence from FLASH to NORMAL puts the flashing cursor on the screen. The segment

XX$ = LEFT$(XX$,LEN(XX$)-(X = 13)-2*(X = 8))

subtracts one character from the end of XX$ if that last character is a carriage return [CHR$(13)], two characters if it is a backspace [CHR$(8)].

If a one-character response is all that is needed, you can simulate a GET command by eliminating the segments that affect XX$ and the statements that refer to A, including the last NEXT. This will put the character in X$.

EXPUT allows use of the left-arrow (BACKSPACE) key but does not allow use of the right-arrow or ESCape functions. A RUN, LOAD, CLEAR, or NEW command will erase the contents of XX$ and the other variables.

Using Make Exec

"Make Exec" (Program 2) is a simple, general-purpose text-entry program. The familiar Apple editing features (right-arrow, left-arrow, and pure cursor moves via the ESCape key) are available. Tap the space bar twice instead of once to get out of the ESCape functions. To back up to a previous line, type CTRL-B. To go forward one line, without changing the contents of the current line, enter a RETURN as the first character on the line. When you have finished entering all of the text, enter a ! as the first character on a new line; you will be prompted for the name the file is to be saved under. If you want to resume editing, don't enter a file name, just press RETURN. If you want to exit the program, type CTRL-C.

Most of EXPUT is automatically entered on the current line when you type CTRL-I. You must designate the contents of PR$, which is used as the prompt, and if desired, use HOME and VTAB before typing CTRL-I. Keep in mind that EXPUT is long, and lines cannot exceed 255 characters. To eliminate a chance of syntax errors, EXPUT begins with a colon.

Program 1.

```
  THE FILE 'EXPUT' CONSISTS OF TWO
LINES.  THEY ARE BROKEN INTO SEGMENTS
IN THIS LISTING FOR CLARITY.
  THE FIRST LINE GIVES THE PROMPT AND
TAKES THE INPUT.  THE SECOND EXECUTES
A COMMAND USING THE INPUT AS A
PARAMETER.  (IF THE FOR-NEXT
LOOPS ARE NOT ON THE SAME LINE THEY
WILL NOT BE EXECUTED.)

XX$="":
HOME:
VTAB 15:
?"ENTER FILE TO BE RUN: ";:
   FOR A = 0 TO 1:
   FLASH:
   ?" ";CHR$(8);:
   NORMAL:
   POKE-16368,0:
      FOR B = 0 TO 1:
      X=PEEK(-16384):
      B=(X>127):
      NEXT:
   X=X-128:
   X$=CHR$(X):
    ?X$;:
    XX$=XX$+X$:
    A=(X=13):
    XX$=LEFT$(XX$,LEN(XX$)-(X=13)-2*(X=8)):
     NEXT
PRINT CHR$(4);"RUN ";XX$
```

Program 2.

```
110  VTAB 1: INVERSE : INPUT "CLEAR SCREEN?
     (Y/N) ";A$: NORMAL : IF  LEFT$ (A$,1) =
     "Y" THEN  HOME
120  VTAB 5
130  DIM C$(100)
140  GOTO 320
150  REM   GET EACH LETTER
160  GET A$: PRINT A$;
170  IF A$ =  CHR$ (13) AND  LEN (B$) = 0 THEN
     A = A + 1: CALL  - 958: PRINT : PRINT A
     ;" ";C$(A);: FOR B = 0 TO  LEN (C$(A)):
      PRINT  CHR$ (8);: NEXT : PRINT " ";:
     GOTO 160: REM  GO FORWARD ONE LINE
180  IF A$ =  CHR$ (13) THEN  CALL  - 958:
     GOTO 300: REM  RETURN
190  IF A$ =  CHR$ (8) AND  LEN (B$) < 2 THEN
     B$ = "": GOTO 160: REM BACKSPACE IF LEN
     (B$) <= 1
200  IF A$ =  CHR$ (8) THEN B$ =  LEFT$ (B$,
      LEN (B$) - 1): GOTO 160: REM  BACKSPAC
     E IF LEN(B$)<>1
210  IF A$ =  CHR$ (21) THEN A$ =  CHR$ ( PEEK
     ( PEEK (40) + 256 *  PEEK (41) +  PEEK
     (36))): PRINT A$;: REM  RIGHT-ARROW
220  IF A$ =  CHR$ (27) THEN  CALL  - 721:
     GOTO 160: REM ESCAPE
230  IF A$ =  CHR$ (2) THEN A = A - 1:B$ = C
     $(A): CALL  - 958: PRINT : PRINT A;" ";
     B$;: FOR B = 0 TO  LEN (B$): PRINT  CHR$
     (8);: NEXT : PRINT " ";:B$ = "": GOTO 1
     60: REM   BACK UP  ONE LINE
240  IF A$ =  CHR$ (3) THEN  STOP : GOTO 160
     : REM  CTRL-C
250  IF A$ = "!" THEN 340
260  IF A$ =  CHR$ (9) THEN 500: REM  CTRL-I
270  B$ = B$ + A$
280  GOTO 160
290  REM   STORE A LINE
300 C$(A) = B$
310 B$ = ""
320 A = A + 1: PRINT : PRINT A;" ";
330  GOTO 160
340  REM   SAVE IT ALL
350 D$ =  CHR$ (4)
360  PRINT : PRINT
370  INPUT "WHAT IS THE FILE'S NAME? ";FL$
380  IF FL$ = "" THEN 160: REM   NULL
390  PRINT : PRINT "SAVING ";FL$
400  PRINT D$;"OPEN";FL$
410  PRINT D$;"DELETE";FL$
420  PRINT D$;"OPEN";FL$
430  PRINT D$;"WRITE";FL$
440  FOR B = 1 TO A
450  PRINT C$(B)
460  NEXT
470  PRINT D$;"CLOSE";FL$
480  END
490  REM   CTRL-I CALLS EXPUT
500 A$ = ":XX$=" +  CHR$ (34) +  CHR$ (34) +
     ":?PR$;:FOR A=0 TO 1:FLASH:?CHR$(32);CH
     R$(8);:NORMAL:POKE-16368,0:FOR B=0 TO 1
     :X=PEEK(-16384):"
510 A$ = A$ + "B=(X>127):NEXT:X=X-128:X$=CHR
     $(X):?X$;:XX$=XX$+X$:A=(X=13):XX$=LEFT$
     (XX$,LEN(XX$)-(X=13)-2*(X=8)):NEXT"
520  PRINT A$;
530 B$ = B$ + A$
540  GOTO 160
```

©

Atari Menu Printer

Dana Noonan

Alphabetize and then automatically print a listing of programs on an Atari disk. It also provides an easy way to quickly update file listings.

Ever wondered just what was on a particular disk? Tried using a disk library program, but gave up because of the time involved in updating your library listing? Do you want a simple way to know what is on all your Atari disks?

For the last six months most of my disks have included some sort of menu program I saw in a computer magazine. These menu programs list all the files on the disk as they appear in the DOS disk directory. Most allow you to call DOS or to run programs by typing a number. While useful, I have never been completely satisfied with any of them.

What annoyed me most was that the programs were listed in a disorganized manner. Since I use word processing and spread-sheet programs extensively, I usually back up each file, using the same title but a different extender. I may have files called CHAPTER2.S12, CHAPTER2.S15, and CHAPTER2.BAC on the same disk. The DOS directory is organized by the order in which disk space is used, and similar file names can be overlooked easily.

Lines 100 to 860 are a simple alphabetized menu program. If your disk contains more than 32 programs, the menu program gives you a choice of viewing either the first or the last 32 programs on a disk. You can call DOS or run any BASIC program which has been previously SAVEd from the menu.

Although I was pleased with the improved appearance of the menu listing, I was still not satisfied with it. What was really needed was a program that would automatically create a printed listing of the programs on a disk (lines 900-1130). I wanted a program that could print either a 4" x 1½" pressure-sensitive label or a slip of paper that I could insert in the disk jacket.

The label could be applied to closed disks – those that were full of programs that I intended to keep indefinitely. I use the simplified Menu (lines 100 to 860) on these disks. For disks that are only partially full or are still being changed, I use a 4⅝" x 5⅛" paper label which slips into the disk jacket.

This is ideal for using with a word processor or spread-sheet. After each session in which you add to, or delete anything from, a disk, simply run the Menu Printer and insert the new listing into the jacket. The list as programmed here is to be printed on an Epson MX-80 printer with Graphtrax, but could be adapted to any printer with a condensed font.

Unless a disk already boots another program automatically, you could use the program in the article, "Automate Your Atari," in the January 1983 **COMPUTE!**.

After formatting a disk, use a pattern disk to duplicate (DOS option J) DOS.SYS, DUP.SYS, MENU and the AUTORUN.SYS.

You could even keep a copy of Menu Printer and DOS.SYS (but not DUP.SYS) on word processor and spread-sheet data disks. Although these programs take about ten percent of the disk space, the ease of generating a hard copy listing of working files is worth it. One possible disadvantage of auto-booting this program: it takes about 18 seconds to bring up the menu, while booting DOS takes only about nine seconds.

The program has significantly improved my ability to find the programs I need quickly and easily. The alphabetized printed list of disk files is particularly useful for finding data files for commercial word processing, data base, and spread-sheet programs.

Menu Printer Listing

Lines	Function
100-150	Dimension the strings. Line 150 names the disk.
200-290	Read the disk directory and set up the string to be sorted.
300-370	The actual string sort.
400-630	Display the program names and enable you to run a program, call DOS, or print a menu.
620-630	Let you switch back and forth between the first and last 32 programs.
700-860	Run the program, if it is a SAVEd BASIC file.
900-1050	Print an alphabetized list of the programs on the disk. Line 920 provides blank fields if they are needed. If you want to add more information, such as a date or your name, to the title line, delete the final ? #4;CHR$(13) from line 980 and add line 985 ? #4;"My Name";: ? #4;CHR$(13). If a pressure-sensitive label is needed, change line

1030 to IF PG<7 THEN GOTO 1020. This works best if the disk contains fewer than 24 programs.

1100-1130 Trap any disk or printer errors. After you check the disk or printer, the program continues.

```
100 REM SET-UP
110 REM SAVE"D:MENU
120 OPEN #2,4,0,"K:":GRAPHICS 0:POKE
    752,1:POKE 559,0
130 DIM A$(900),P$(15),S$(13),B$(15),
    BL$(40),F$(15),L$(13),N$(30),Z$(1
    6):Z$="AND PRESS RETURN"
140 A$(1)=" ":A$(900)=" ":A$(2)=A$:P$
    ="":S$="":B$="":BL$=" ":BL$(40)=B
    L$:BL$(2)=BL$:F$=""
150 N$="JOURNAL AND WORKING FILES"
200 REM READ DIRECTORY
210 TRAP 1130:OPEN #1,6,0,"D:*.*":TRA
    P 40000
220 FOR I=0 TO 14:B$=" ":NEXT I
230 INPUT #1,P$
240 IF P$(5,8)="FREE" THEN GOTO 290
250 P$=P$(3,13)
260 REC=REC+1:CC=LEN(P$)
270 IF CC=0 THEN T=REC:CLOSE #1:GOTO
    300
280 A$(REC*13-12,REC*13-12+CC)=P$:GOT
    O 230
290 F$=P$:CC=0:GOTO 270
300 REM SORT
310 T=INT(T/3)+1:FOR L1=1 TO REC-T:FO
    R L2=L1 TO 1 STEP -T
320 IF A$(L2*13-12,L2*13)<=A$((L2+T)*
    13-12,(L2+T)*13) THEN 360
330 S$=A$(L2*13-12,L2*13):A$(L2*13-12
    ,L2*13)=A$((L2+T)*13-12,(L2+T)*13
    )
340 A$((L2+T)*13-12,(L2+T)*13)=S$
350 NEXT L2
360 NEXT L1
370 IF T>1 THEN 310
400 REM PREVIEW DATA
420 P=1:X=1:TEC=REC-32:IF TEC<=0 THEN
     440
430 IF TEC>0 THEN N=16:REC=32
440 N=INT(REC/2)
450 POKE 559,34:GRAPHICS 0:POKE 752,1
    :POKE 82,2
460 L=LEN(N$):LL=(38-L)/2:POSITION LL
    ,0:? N$
470 POSITION 12,2:? F$;"S  "
480 POSITION 2,4:FOR MX=X TO N:S$=""
490 IF MX<=9 THEN S$=CHR$(32)
500 ? S$;MX;") ";:? A$(MX*13-12,MX*13
    ):NEXT MX
510 POKE 84,4:FOR MX=N+1 TO REC:S$=""
520 POKE 85,20:IF MX<=9 THEN S$=CHR$(
    32)
530 ? S$;MX;") ";:? A$(MX*13-12,MX*13
    ):NEXT MX
540 POSITION 2,21:? "1) RUN
    {7 SPACES}2) DOS{7 SPACES}3) PRIN
    T"
550 IF TEC>0 THEN POSITION 13,22:? "4
    ) NEXT PAGE"
560 POSITION 13,23:? "CHOOSE OPTION";
570 GET #2,R:IF R<49 OR R>52 THEN 560
580 A=VAL(CHR$(R))
590 IF A=1 THEN 700
600 IF A=2 THEN DOS
610 IF A=3 THEN 900
620 IF A=4 AND P=1 THEN ? CHR$(125):R
    EC=REC+TEC:X=33:N=INT(TEC/2)+32:P
    =2:GOTO 460
630 IF A=4 AND P=2 THEN P=1:GOTO 460
700 REM RUN PROGRAM
710 TRAP 710:POSITION 2,21:? "
    {6 SPACES}INPUT PROGRAM YOU WANT
    {6 SPACES}":POSITION 11,22:? Z$;:
    INPUT X:TRAP 40000
720 IF X<>INT(X) THEN 710
730 IF X=0 THEN 710
740 L$=A$(X*13-12,X*13)
750 IF L$(9,9)=" " THEN L$=L$(1,8):GO
    TO 770
760 FOR X=12 TO 10 STEP -1:L$(X,X)=L$
    (X-1,X-1):NEXT X:L$(9,9)="."
770 S$="D:":FOR I=1 TO LEN(L$):IF L$(
    I,I)=" " THEN 790
780 S$(LEN(S$)+1)=L$(I,I)
790 NEXT I
800 POKE 752,1:POSITION 6,22:? "
    {4 SPACES}LOADING ";L$
810 TRAP 820:RUN S$:TRAP 40000
820 POSITION 6,22:? "{3 SPACES}CANNOT
     RUN ";S$:TRAP 40000:FOR WAIT=1 T
    O 900:NEXT WAIT
830 POSITION 2,19:FOR J=20 TO 23
840 ? BL$(1,38);
850 NEXT J
860 POSITION 2,19:GOTO 540
900 REM PRINT
910 GOTO 940
920 REC=REC+1:P$="{11 SPACES}"
930 CC=LEN(P$):A$(REC*13-12,REC*13-12
    +CC)=P$
940 IF REC/4<>INT(REC/4) THEN 920
950 TRAP 1110:OPEN #4,0,8,"P:":TRAP 4
    0000
960 ? #4;CHR$(27);CHR$(68);CHR$(3);CH
    R$(23);CHR$(43);CHR$(63);CHR$(0);
970 ? #4;CHR$(15);"------------------
    ---------------------------------
    ---------------------------"
980 ? #4;CHR$(13);CHR$(9);N$;CHR$(9);
    F$;"S";:? #4;CHR$(13)
990 PG=1
1000 N=INT(REC/4):FOR Q=1 TO N:FOR MX
     =Q TO REC STEP N
1010 ? #4;CHR$(9);A$(MX*13-12,MX*13);
     :NEXT MX:? #4;CHR$(9):PG=PG+1:NE
     XT Q
1020 ? #4:PG=PG+1
1030 IF PG<=27 THEN GOTO 1020
1040 ? #4;"---------------------------
     ---------------------------------
     --------------------"
1050 CLOSE #4:GOTO 540
1100 REM ERROR TRAPS
1110 CLOSE #4:? "{CLEAR}":POKE 82,11:
     POKE 752,1:POSITION 13,10:? "CHE
     CK PRINTER":? "{2 DOWN}";Z$:GET
     #2,R
1120 POKE 82,2:GOTO 400
1130 ? CHR$(125):POKE 82,11:POKE 752,
     1:POSITION 14,10:? "CHECK DISK":
     ? "{2 DOWN}";Z$:GET #2,R:GOTO 20
     0
```

©

LEARN TO TYPE OR GET BLOWN TO BITS.

MasterType™ and Atari™ make typing a blast.

Now anyone can learn to type and have hours of fun at the same time. With MasterType. We've combined the fast-action of blow 'em up video games with the best instructional program available. You either learn to type or the aliens ZAP your spaceship into oblivion. The results? Highly motivated and enjoyable learning.

MasterType earns a ten-gun salute.

MasterType is fast becoming the best selling educational software product designed for use with the Atari 800 home computer.

It's no wonder, when you consider what InfoWorld magazine had to say:

"MasterType is an excellent instructional typing game. We had fun reviewing it, and we highly recommend it to those who want to learn typing in an unconventional but motivating way."

InfoWorld also went on to rate MasterType as "excellent" in all four of its categories.

MasterType teaches your fingers to fly.

Face it. If you're like most people, the most inefficient components on your computer are probably your fingers. But you can change all that with MasterType. In just a few easy lessons your fingers will be doing things you thought them incapable of.

MasterType. With 18 explosive learning levels. You'll either learn to type or get blown to pieces.

$39.95. Requires disk drive 32k and basic.

MasterType

Lightning Software ©
P.O. Box 11725, Palo Alto, CA 94306
(415) 327-3280

Part I:

COLOR COMPUTER GENERAL PURPOSE DATA BASE

Jeffrey S. Yohay

For TRS-80 Color Computer, this tutorial can serve as a model for creating a data base manager. Among the most useful of computer applications, data bases can manipulate and process lists, catalogs, and thousands of other kinds of information. This article concludes next month with a discussion of screen displays, program structure, and the data base program itself.

As a personal information manager, the TRS-80 Color Computer is often overlooked in favor of its more expensive counterparts. This may be because of the machine's somewhat cramped screen format or its calculator-like keyboard.

Whatever the reason, it is a serious mistake to relegate the Color Computer to a back seat in information handling. This is because the Color Computer offers as standard equipment a fast and reliable cassette tape storage system with many disk-like features for data storage. This allows even the lowest priced Color Computer to store and retrieve personal data quickly and efficiently – an important consideration in a machine that is likely to be purchased by computer novices who won't be starting right away with a disk drive.

Using the TRS-80 Color Computer's powerful cassette system, this program is a model for a personal information management system. It is called the "Videotape Movie Data Base Program" (VMDP), because it was designed to catalog and manage a collection of movies on videotape. While the VMDP is designed around the Color Computer's powerful cassette storage format, it also makes good use of the limited 16 line x 32 character text screen to display a great deal of information about the cataloged movies. The program is written in Extended Color BASIC and requires at least 16K of RAM.

As more and more video enthusiasts are discovering, the TRS-80 Color Computer is a good microcomputer to include as part of a home video system. It is easily connected to any television or videorecorder, and it combines all the features of a ROM-based video game system and a powerful microcomputer system. The high-resolution color graphics can be used to produce spectacular video displays for games and simulations. Those same graphics, combined with the Color Computer's innovative cassette-to-television audio channel, can be used for numerous educational applications. And the CPU itself, the Motorola 6809, is a powerful one that can support many applications.

How The VMDP Can Help

Let's take a look at a typical video enthusiast's collection of movies on videotape. Chances are that the joy of possessing a recording of some rare old movie will quickly fade the first time the intrepid video buff goes to find the recording and can't! Scribbled labels, out-of-order index cards, and frazzled nerves are typical of a videotape cataloging system. Yet those video hobbyists who are organized enough to develop an accurate and useful catalog of their videotapes can find themselves spending as much time on the recordkeeping as on the collecting.

The VMDP offers a cure to these ills. For each movie in your collection, the VMDP will allow you to store:

1. The title.
2. The year of release.
3. The type of movie (comedy, mystery, etc.).
4. The name of the director.
5. The names of up to three actors/actresses.
6. The videotape you recorded it on.
7. The location on videotape by VCR counter number.
8. The recording speed.
9. The approximate viewing time.
10. The time remaining on the videotape.
11. The date you recorded it.
12. The channel you recorded it from.
13. Whether the movie is in color or black-and-white.
14. Whether you recorded it with or without commercials.

Once you've cataloged your movies, the VMDP will let you:

1. Display all data for any movie or movies.
2. Display all movies by title and type.
3. Search for and display data for a particular movie.
4. Print all movie data or only title and type.
5. Enter data for a new movie.
6. Delete data for an existing movie.
7. Sort the movie data by title, type, or videotape number.

Using the VMDP, your Color Computer can organize even the most haphazard collection of videotape movies and let you choose the movie you're in the mood to see, when you're in the mood to see it. Feel a little low? Just have the VMDP sort your movie collection by type, and pick out a good comedy. Or let the VMDP search through your collection for that particular movie you haven't seen in months. Finding entertainment to suit your mood couldn't be easier.

Data Storage Concepts

The most important consideration in the design of the VMDP was the Color Computer's cassette tape storage system. Before I tell you how I decided on the VMDP's tape data storage format, let's take a look at how this cassette system works.

For those of you who are new to data storage concepts, let me first define a few terms. Data is stored on a magnetic device (cassette tape or floppy disk) in groups of related information called "files." Files themselves are collections of related data items called "records," which are usually the smallest units of data read or written to a cassette tape or disk at one time. Within each record, the data is further organized into "fields." This is done so that once you read the record from the tape or disk into the computer, you'll know where to find any particular subset of data that you're looking for.

If all you want from the record is the title of the movie being described, knowing the location of the field containing the title makes it easy to find. Remember, too, that though the computer handles the storage of records and files, the storage of the data within the records is completely up to you. So it's a good idea to do what the computer does, and organize your data fields so you'll always know how to get your data back quickly and easily.

Files can be stored by the computer in two different ways: sequential-access or random-access files. For tape storage, however, we can use only sequential-access files. These have to be read or saved one record after another; thus, you can reach a record in the middle of a file only by reading in all the records that come before it. Random-access files have records that can be read or saved no matter where they are in the file, but that can only be done if you have a disk.

The biggest advantage of random-access is that you don't have to read an entire file into memory before you start extracting the information you want from it. Just read in the records that you want and get the data out of the fields in those records. Of course, you could do that with sequential-access files, but you'd have to go through the entire file every time you wanted a particular group of records. For cassette files, that would be too slow to be practical.

The Color Computer Cassette

The Color Computer's cassette tape system has many improvements over those of earlier TRS-80 models. The most important of these improvements is the speed at which programs and data are saved on tape: 1500 baud (bits per second), or about 11,000 characters per minute. There are also many disk-like features; one of these is the use of a file structure for all data stored on cassette. Instead of simply PRINTing data to cassette directly from a variable as in the Model I/III, the Color Computer opens a file on tape, stores the data, then closes the file when you're done. This is the same way sequential-access data files are stored on disk.

Central to this data file storage method is the use of "buffers." A file buffer is an area of RAM memory reserved for data that is to be read from, or written to, cassette. When you want to read or write cassette data, you use the Color Computer's OPEN command to initialize a file buffer in memory. This buffer is used to hold cassette data during cassette I/O operations.

When you have data to save on tape, the buffer is filled before the data is written to tape, keeping data transfer time to a minimum since the tape doesn't have to be moved for every variable value that your program tells the computer to save. This also enables data to be stored on tape very efficiently, since the computer "saves it up" until it can write one buffer's worth of data to tape. Similar use is made of the file buffer when loading data (i.e., assigning tape data to a variable) to minimize tape movement and data transfer time.

Using the OPEN command when saving data also causes BASIC to write a block of data onto tape called the "NAMEFILE" block. (These block names and descriptions are all from the Radio Shack *TRS-80 Color Computer Technical Reference Manual.*) The NAMEFILE block consists of 15 bytes of BASIC-generated data that describe your data file in several ways, including the name of the file and how it is recorded.

OPENing a data file to load data tells BASIC to read this NAMEFILE block to see if this is the data file you wanted; if not, the cassette will be searched until the file you wanted is found. This lets you store several independent data files (perhaps collections of movies by certain directors or with certain actors/actresses) on the same tape.

Similarly, use of the CLOSE command tells BASIC to delete the file buffer and write an "END-OF-FILE" block to tape. When reading in the data file, BASIC can then use the END-OF-FILE block to tell when it has reached the end of your data file. The EOF(-1) function will be "true" (equal to -1) if the END-OF-FILE block has been reached; use it when loading a data file to check whether all of the data you wanted has been read in from tape.

In between the NAMEFILE and END-OF-FILE blocks are your actual data. These are stored in "DATA" blocks that both describe the data and contain up to 255 bytes of the data itself.

Building Your Data Base

Using my knowledge of the Color Computer cassette system, I decided to store the data for each movie in 127-byte records. This would allow my movie data to make the best use of the 255-byte tape data blocks, and would make it simple for BASIC to read and write data to and from the tape buffer and the tape itself. And, since Color Disk BASIC stores data on disk in multiples of 256 bytes, I would get the added benefit of movie data that could be easily adapted to a random-access disk system in the future.

Once I had chosen the 127-byte length for the movie records, I decided that this would be the record size no matter how few bytes were actually required to describe a particular movie. This "fixed length" record format has a big advantage over "variable length" records (where each record is only as long as required to describe each movie adequately).

Though it takes more tape to store a file with records that may be filled with a lot of blanks, a fixed record length insures that the locations of all movie data within a record (the fields) are the same in every record. This makes it easy to get the data for displaying, printing, and sorting. And no matter what the future brings (e.g., a disk drive!), my videotape data would be consistent and easily accessed by any program on cassette or disk.

I then had to decide how to place the movie data within the record. From experience with a pencil-and-paper system, I knew what information I wanted to have for each movie, and I could see that I would need 17 individual fields within each record to store this information. So, after deciding on the size of each field and its location within the record, I came up with the record format illustrated in Table 1 next month.

I chose the size of each field so that each was just large enough for the data it was to contain, but not so large that it would be filled with blanks most of the time (because of the fixed record length). This was particularly important for the director and actor/actress fields, where the names could vary widely in length. Since my favorite director's name (Alfred Hitchcock) is 16 characters long, I thought that would be a good length to start with. As it turned out, that length was ideal, and left more than enough bytes for the rest of the data, with two bytes left over for future use (they're filled with a slash "/" for now). Note that some fields had to be only long enough for a one-byte or two-byte code that the VMDP can recognize and expand into usable information.

There was a method to my madness in the ordering of the fields within the record, too. If you BREAK the program after loading data and then PRINT some values from the movie record array R$, each 127-byte record will fit neatly on four of the Color Computer's 32-character screen lines. The title and year will be on the first line, the director and actors/actresses will be on the second and third lines, and the remaining information will be on the fourth line, separated for easier readability by the slashes I placed in the "future use" fields. This bit of clever record formatting makes it easy to check the contents of any movie record in the R$ array.

However, my real purpose in organizing the records to fit neatly onto the Color Computer screen was to make it easier to enter data into the R$ array directly. This can be a real boon to those who already have large videotape movie collections who want a fast way to enter numerous movie records into the VMDP without repeatedly running the "add record" routine.

For example, if N is the number of movie records, then just set R$(N + 1) equal to the four lines of movie data to enter a new record into the movie array directly (a good screen editor, such as Datasoft's *S.E.C.S.*, makes this a lot easier). Do this in "command mode" (i.e., without a line number), and then restart the program with "GOTO 50" to avoid the CLEAR statement in line 40. (Note that because a RUN statement contains its own CLEAR, you must use a GOTO when you want to restart the program with your data intact.) ©

Apple Subroutine Capture

R. W. W. Taylor and Max Hailperin

Do you include certain favorite BASIC subroutines in program after program? The easiest way to incorporate a standard subroutine into a new program is to EXEC the code from an existing text file, as explained on page 76 of the *Apple II DOS Manual*. A short program is given on that page for "capturing" specified lines from a program already in memory and writing the lines as text to a sequential file for later retrieval by an EXEC command.

The main inconvenience of this particular approach is that the capture subroutine must be typed in new each time it is to be used, with details specific to the situation at hand.

This nuisance can be avoided. In fact, it is possible to create and store a master file Capture so that a user who simply types EXEC CAPTURE will be interrogated about the desired file name and line-number range, and the desired capture will then be performed without any further action by the user.

The text of Capture appears in Program 1. This text can be entered into a file by a program such as File Builder (Program 2). Note the subroutine at line 8000. The purpose of this subroutine is to allow input of arbitrary text strings, including commas, colons, and hyphens. It is a good example of the sort of subroutine that is handy to capture and maintain for re-use in other programs.

Saving To Memory

Once Capture has been stored on disk, and a program containing lines to be captured has been loaded or created, the command EXEC CAPTURE is issued. The first effect is to overlay lines 1-18 of the program in memory – lines in this range cannot be captured. These lines are then run by the RUN at the end of Capture. The user is asked to specify a name for the file to be created and two line numbers indicating the range of code to be captured. The line numbers must be entered separated by a *comma*.

The program then proceeds to build a file called Tempcapture, incorporating the information supplied by the user. Before ending, the program issues a command to EXEC TEMPCAPTURE. Once again, lines in the range 1-18 are overlaid, and the new lines are run. This time, the desired capture is performed, Tempcapture is deleted, and the completion of the task is announced.

Note that if the user's disk already happens to contain a text file named Tempcapture, this file will be overwritten and then deleted. An already existing text file will also be overwritten if its name is specified as the file to be created. However, if the name specified represents an existing binary, Applesoft, or integer file, a "FILE TYPE MISMATCH" message will be generated, and the process will halt without any damage to the file.

Program 1: Text For Capture File

```
1  REM  - CAPTURE SUBROUTINE
2 CD$ =  CHR$ (4): REM   CONTROL D
3 Q$ =  CHR$ (162): REM   QUOTE CHARACTER
4  HOME : INPUT "FILE NAME TO BE CREATED? ";F$
5  VTAB 4: INPUT "LINES TO BE CAPTURED? ";LO
   %,L1%
6  PRINT CD$;"OPEN TEMPCAPTURE"
7  PRINT CD$;"WRITE TEMPCAPTURE"
8  PRINT "4 PRINT CD$;";Q$;"OPEN ";F$ + Q$
9  PRINT "5 PRINT CD$;";Q$;"WRITE ";F$ + Q$
10  PRINT "6 LIST ";LO%;"-";L1%
11  PRINT "7 PRINT CD$;";Q$;"CLOSE ";F$ + Q$
12  PRINT "8 PRINT CD$;";Q$;"DELETE TEMPCAPT
    URE";Q$
13  PRINT "9 HOME: PRINT ";Q$;"FILE ";F$;" H
    AS BEEN CREATED.";Q$
14  PRINT "10 END"
15  PRINT "RUN"
16  PRINT CD$;"CLOSE TEMPCAPTURE"
17  PRINT CD$;"EXEC TEMP CAPTURE"
18  END
```

Program 2: EXEC File Builder

```
10  REM  ** FILE BUILDER **
20 CD$ =  CHR$ (4): REM  CONTROL D
30  HOME : PRINT "ENTER NAME OF FILE TO BE B
    UILT:"
40  PRINT : HTAB 10: INPUT F$: HTAB 10: VTAB
     PEEK (37): PRINT " "
50  PRINT : PRINT "INPUT LINES ONE BY ONE."
60  PRINT "TO END, JUST PRESS RETURN."
70  VTAB 9: POKE 34,8: REM   SET TOP OF TEXT
     WINDOW
80  PRINT CD$;"OPEN ";F$: PRINT CD$;"DELETE
    ";F$: PRINT CD$;"OPEN ";F$
90  FOR I = 0 TO 1
100  PRINT "* ";: GOSUB 8000
110  IF 0 <  LEN (IN$) THEN I = 0: PRINT CD$
    ;"WRITE ";F$: PRINT IN$: PRINT CD$
120  NEXT I
130  PRINT CD$;"CLOSE";F$
140  HOME : POKE 34,0: REM   RESET TEXT WINDOW
150  PRINT "* FILE ";F$;" HAS BEEN BUILT."
160  END
8000  CALL 54572: REM  INPUT SUBROUTINE
8010  FOR B = 512 TO 751
8020  IF  PEEK (B) <  > 0 THEN  NEXT
8030 IN$ = ""
8040  POKE  PEEK (131) + 256 *  PEEK (132) +
    1,0
8050  POKE  PEEK (131) + 256 *  PEEK (132) +
    2,2
8060  POKE  PEEK (131) + 256 *  PEEK (132),B
     - 512
8070 IN$ =  MID$ (IN$,1)
8080  RETURN
```

©

Part II

Commodore 64 Video – A Guided Tour

Jim Butterfield, Associate Editor

We now continue our guided tour of the video capabilities of the Commodore 64 computer. Along the way we'll stop for lots of experiments, things for you to type in and watch the effects of manipulating this remarkably versatile computer.

The story so far: we're touring the 6566 chip, which gives the Commodore 64 its video. We noted last month that the chip goes to memory for its video information, but can only reach 16K; the computer controls which 16K bank via control lines in 56576 (hex DD00). Then we picked out the functions of the video control word at 53265 (hex D011).

We've seen the variety of important controls that we can reach in location 53625: vertical screen positioning, screen blank, bit mapping, and extended color. There's a second control location, at 53270 (hexadecimal D016); let's look at it.

The first thing we should note about this location is that the two high bits are not used. That means that we can usefully POKE only values from 0 to 63 in there. It happens that if we PEEK 53270, we'll probably see a number that is 192 too big; if you want to see the working value, use PEEK(53270) AND 63, which will throw away the unused part of the number.

We saw a vertical fine scroll in location 53265. Location 53270 has a horizontal fine scroll that works exactly the same way. Type:

```
FOR J=8 TO 15:POKE 53270,J:NEXT J
```

You'll see the screen characters slide over horizontally. As with the vertical fine scroll, we also have facilities for trimming the size of the screen. Restore the screen to its original form with POKE 53270,8. Then shrink the screen by typing POKE 53270,0. You'll see a character disappear from each end. In other words, you now have a 38-character screen instead of 40 characters. Don't forget that fine scroll and shrink can be used effectively together.

If you add 16 to the contents of 53270, you'll switch to multicolor mode. This is not the same as extended color which we discussed previously. Multicolor allows *selected* characters to be shown on the screen in a combination of colors. Extended color, you may remember, allows screen background and foreground to be set individually for each character.

If you're familiar with the VIC-20, you'll find that setting the multicolor mode makes the Commodore 64 behave in the same way. Here's the trick: we invoke multicolor on an individual character by giving that character a color value greater than 7. This way, the regular colors (red, blue, black) behave normally, but the new pastels (grey, puce) switch to multicolor mode.

You'll need to create a new character base to exploit the advantages of multicolor, since the old characters weren't drawn with color in mind. However, we can get a quick idea of the feature by invoking it: POKE 53270,24 sets up multicolor; the screen characters may turn a little muddy, but don't worry about them. Set a primary color such as cyan and type a line. Normal, right?

Next, set up one of the alternate colors (hold down the "Commodore" key and press a key from 1 to 8). Type some more; you'll get multicolor characters. They won't make much sense, since the character generator isn't building the colors suitably; but you can see that something new is going on.

Adding 32 to the contents of 53270 gives chip reset. You won't want to do this very often – it's done on your behalf when you turn the power on. If you do use chip reset, remember that to make it work, you must turn reset on and then off again. POKE 53270,32:POKE 53270,8 will clear you out of multicolor mode.

Setting Screen And Characters

Location 53272 sets the location of screen RAM (the video matrix) and the character generator (the character base). Don't forget that they must be in the same 16K block, as determined by the

SJB DISTRIBUTORS. THE MOST COMPETITIVE PRICES ON COMMODORE.

commodore

MONITORS--GREAT RESOLUTION (64 or VIC)

Panasonic 13" 1320V (Also a great color TV)	$489
Amdek Color I	329
Nec JB 1201M, 12" Color	329
Nec JB 1201, 12" Green Phosphor	159
Amdek Video 300L	159
Transtar (High Resolution)	143
Video/Audio Cable	25

Arcade Joysticks--Heavy duty with 2 firing buttons! Great for the VIC or 64 .... $25

SuperPET (5 languages, 2 processors)	$1409
CBM 8032 Computer, 80 column	1029
CBM Memory Expansion, 64K	359
PET 4032, 40 column	950
CBM 8050, 1 Mg, Dual Drive	1259
CBM D9060, 5 Mg, Hard Disk	2240
CBM D9090, 7.5 Mg, Hard Disk	2600
CBM 4040, 340K Dual Drive	919
CBM 2031, 170K Single Drive	489
DC Hayes Smart Modem	220

PRINTERS--LETTER QUALITY

CBM 8300, 40cps	$1450
Diablo 620, 25cps	995
Nec Spinwriter 7700, 55cps	2350
Nec Spinwriter 3500, 35cps	1600

PRINTERS--DOT MATRIX

CBM 4022, 80cps/graphics	$395
CBM 8023, 150cps/graphics	599
Okidata 82A, 120cps/serial or par	449
Nec 8023A (parallel)	499
Epson MX 50 F+	529
IDS Microprism	539

BUSINESS SOFTWARE

WordPro 4+ or 5+	309
Administrator (Awesome Database)	489
VisiCalc (expanded)	199
The Manager (database)	199
Legal Time Accounting	400
BPI A/R, G/L, Job Cost, Inventory, Payroll	325 pkg.

SJB will service any VIC or CBM64.

MasterCard, Visa, Money Order, Bank Check
COD (add $5) accepted.
Add 3% surcharge for credit cards.
In stock items shipped within 48 hours, F.O.B, Dallas, TX
All products shipped with manufacturer's warranty.

TO ORDER CALL TOLL FREE
800-527-4893 800-442-1048 (Within Texas)
SJB will meet any competitive price under similar in-stock conditions.

SOFTWARE FOR CBM 64

Word Processing (Word, Palabra)	$85
WordPac	70
COCO (great computer tutoring game)	44
COCO II (build your own games easily)	49
Home Finance Package	25
General Ledger, A/R, Inv.	175 pkg.
CBM EasyCalc	90
CBM Easy Finance	50
CBM Easy Schedule	80
Data Manager	70
Pet Emulator (emulates 4.0 basic)	30
Sprite-Magic (build sprites on screen with Joystick, save to disk or cassette)	$19
Assembler Package for CBM 64 (cassette) Editor (creates and updates source code) Assembler, Loader, Disassembler	50
Mail Mate	45
IEEE Interface (64)	95
Parallel Interface (Epson, Okidata, IDS, Nec)	80
RS232 Printer Interface (Okidata etc.)	65
Apple Loader	100
Programming Reference Guide	18

VIC PRODUCTS

VIC 20 Computer, 5K	$179
VIC Datasette Recorder	60
VIC 1541 Disk Drive	395
VIC MODEM (for CBM 64)	95
VIC 1525 Graphic Printer (for CBM 64)	325
8K Memory Expansion Cartridge	49
16K RAM	99
24K RAM	155
IEEE Interface (VIC)	85
VIC 3 slot Expander	43
VIC 6 slot Expander	83
RS232 Printer Interface	65
Cassette Interface	27
Intro to Basic I or II	23
Home Finance Package	48
Turtle Graphics	35
Heswriter (great wp)	35

VIC GAMES

Choplifter (cartridge)	37
Apple Panic	37
Trashman	37
Blackhole	37
Sargon II	30
Satellites & Meteorites	38
Seawolf	23
Cosmic Cruncher	23

SJB DISTRIBUTORS, INC.
10520 Plano Road, Suite 206
Dallas, Texas 75238
(214) 343-1328

Business Hours
M-F 8 to 6
Sat. 10-2

Prices are subject to change without notice.

low bits of address 56576.

You can get the BASIC address of screen RAM in this way: take the contents of 53272 and divide by 16; then throw away the remainder and multiply by 1024, and you have the screen address. You can get the BASIC address of the character base in this way: take the contents of 53272 and divide by 16. Then take the remainder, subtracting one if it's odd, and multiply by 1024; that's the character base address. Both addresses will need to be adjusted to allow for the 16K quadrant we have selected.

Now: if we are in bit map mode, we get the character base address in a slightly different way: divide the contents of 53272 by 16; take the remainder and divide by 8, discarding the remainder; finally, multiply by 8192. That's the bit image; it should be either 0 or 8192.

How does this work out in the standard Commodore 64? We may PEEK 53272 and see a value of 21. That means the screen is at INT(21/16)* 1024, or address 1024. Right on target. The character matrix works out: the remainder of 21/16 is 5, so drop one for the odd number, giving 4; multiply by 1024 to get address 4096. You may remember that our discussion last month indicated that RAM was replaced by the character generator ROM at this video chip address. And when we flipped to bit mapping in the last episode, we still got remainder 5; divide by 8 giving 0, then multiply by 8192 – you still get 01 high resolution screen from address 0.

If you'd like to try your hand at the arithmetic, flip to upper-/lowercase mode (hold down SHIFT and press the Commodore key) and see what addresses have changed. Or if you'd rather, try typing in FOR J=1 TO 100:POKE 53272,21:POKE 53272,23:NEXT J and watch the action.

The Raster Register

Location 53266 (hex D012) and the high bit of the previous location are not of much use to the BASIC programmer, but can be very valuable to the machine language tyro. Here's the idea: by looking at these locations, you can tell exactly where the screen is being scanned at that moment. This allows you to change the screen as it's being scanned: halfway down, you could switch from characters to bit map, or change to multicolor, or move a sprite that has already been displayed.

If you're really interested in machine language, you may want to take an extra step: instead of watching where the screen is, you can leave the message "Wake me when you get to scan line 100." ML tyros will recognize this as an interrupt request. How do you set the identity of the desired scan line? By placing it into the same locations, that's how. We have a dual function here: when we read, we recall the scan location; when we write, we store an interrupt value.

Light Pen

Locations 53267 and 53268 (hex D013 and D014) are the light pen registers. An Atari-style light pen can be plugged into joystick port number one; if it sees a suitable signal from the screen, the X and Y values will be latched into these registers. The light pen can be used on an interrupt basis: we can "stop the music" and get immediate action if we choose to set things up that way.

This is the second time we've mentioned interrupts; perhaps we'd better discuss them a little more closely.

Interrupts

Interrupts are for machine language experts – things happen too fast for BASIC to cope in this area. There are four types of interrupts: raster, light pen, and two kinds of sprite collision. (We'll talk about sprites in Part III next month.) We may use all of them or none; and even when these signals are not used for interrupt, we can check them.

Location 53273 (hex D019) tells us which of the four events has occurred. We don't need to make the interrupts "live"; they will signal us any time the particular event happens. The weights are as follows:

1 (bit 0) – the raster has matched the preset line value;

2 (bit 1) – a sprite has collided with the screen background;

4 (bit 2) – a sprite has collided with another sprite;

8 (bit 3) – the light pen has sensed a signal;
128 (bit 7) – one of the above has triggered a live interrupt.

Once any of the above takes place, the bit will remain stuck on until you turn it off. How do you turn it off? This may sound goofy, but you turn an interrupt signal off by trying to turn it on. Hmmm, let me try that again. Suppose that we have both a raster and a light pen signal; we'll see a value of 9 (8+1) in the interrupt register. Now suppose further that we are ready to handle the light pen, so we want to turn its signal off. We do this by storing 8 into location 53273. Huh? wouldn't that turn it on? Nope, it turns it off, and leaves the other bit alone. So after storing 8, we look at the register again, and (you guessed it) we see a value of 1 there. Honest.

Location 53274 (hex D01A) is the interrupt enable register: it sets the above signals for "live interrupt." Select bits 0 to 3 corresponding to the interrupts you want. Whatever live interrupt you select will now trigger a processor interrupt and also light up that high bit of 53273. Don't forget to

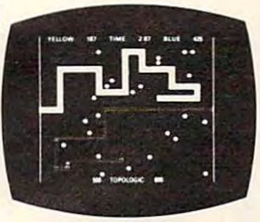
ARCADE GAMES!

NEW FOR THE

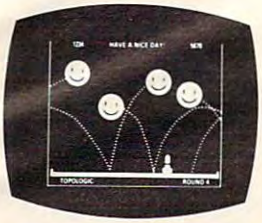
commodore **64**

COLLISION

Avoid the walls, the purple dots, and the lines as you maneuver to make your opponent crash before time runs out. 9 levels of difficulty. Bonus time for high scores. 1 or 2 players. Joysticks required.



AT YOUR DEALER NOW!



HAVE A NICE DAY!

Move your base and fire your missiles to blast Happy Faces from the sky! Full sprite graphics! 1 or 2 players. Keyboard, joystick, or paddle controls.

AVAILABLE SOON!

All games compiled from BASIC for speed. Ask for COLLISION! at your dealer, or send $12.95 for each program on cassette tape, $15.95 on disk, plus $1.50 shipping.

Dealer inquiries invited.

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

Topologic

810 North Ninth Street
Burlington, Iowa 52601

PET/CBM PROVINCIAL PAYROLL

Wycor Business Systems has developed a complete payroll system for Canada.

• Set up files for over 200 employees • Calculate and print payroll journal • Print cheques • Print monthly submission for Revenue Canada • Accumulate and print T-4s • Complete employee lists.

This system comes with full user documentation and tutorial disk.

Complete System	**$850.00**
Manual only	**25.00**

Call collect (416) 444-3492 for information or contact your dealer.

WYCOR BUSINESS SYSTEMS LIMITED
170 THE DONWAY WEST, STE. 401,
DON MILLS, ONT. M3C 2G3

64 ** ALL NEW!!! ** 64

— — — * * * — — —

SOFTWARE FOR COMMODORE 64

WORD-PAC $74.95
- Print up to 99 pages of text.
- Automatic tabbing/Centering/Underlining.
- Copy Lines/Merge/Plus More!
- Coded in Machine language.

CALC-PAC $74.95
- Interface-Compatible with WORD-PAC & DATA-PAC.
- Coded in our own Unique Spread-Sheet language.
- User-Friendly Mathematical Applications.

DATA-PAC $39.95
- Interface-Compatible with WORD-PAC & CALC-PAC.
- User defined Formats/Search & Sorts.
- Printer compatable.

EDITOR-PAC $69.95
- Complete Programmer's Editor.
- Auto-Number/Renumber including goto & gosub.
- Program Merge/Global Search and Replace.
- Plus Much More!

ASSEMBLER-PAC $59.95
- Programmers take note!
- Mnemonic format to Machine Language.
- Link Modules/External references, More!

HOME-ACCOUNTANT $29.95
- Checkbook with reconciliation routine.
- Hard-Copy listing option.
- Search and Review/Chart of Accounts.
- Income and Expense.

ANNOUNCING . . .

The PCS/8064 Upgrade Module for the 64

On power-up the PCS/8064 provides:
- 80-column video output.
- WORD-PAC word processing.
- CALC-PAC spread sheet mathematics.
- DATA-PAC data base system.
- Exit to BASIC.
- All Applications Interface-Compatible.

Check local dealers or Call Pacific Coast Software for retail pricing.

Plus Full line of Games/Home Software for 64
Free Catalog Offer.

PACIFIC COAST SOFTWARE

3220 S. Brea Canyon Rd.
Diamond Bar, CA 91765
(714) 594-8210

218 S. Main/Box 147
LeSueur, MN 56058
(612) 665-6724

Mid-Eastern Distribution:

PERIPHERALS PLUS .... (215) 687-8540
155 E. Lancaster Ave. - Wayne, Penn. 19087

New England Distribution:

OMICRON .... (617) 769-6867
1416 Providence Highway - Norwood, Mass. 02062

Dealer Inquires Encouraged.

Commodore 64 and 64 are trademarks of Commodore Business Machines.

Table 1:

6566 Video Chip
C64 Control and Miscellaneous Registers

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	Back	Raster	53274

Color Registers

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	
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

shut the interrupt flag off when you service the interrupt, using the method indicated in the previous paragraph. Otherwise, when you finish the job and return from the interrupt (with RTI), it will re-interrupt you all over again.

A Little Color

Some of the colors we have mentioned and some we have yet to discuss are neatly stored in addresses 53280 to 53286 (hex D020 to D026). We may store only values 0 to 15 here, for the 16 Commodore 64 colors.

The chart shows it all: the exterior (border) color; then four background colors (they may be selected as part of multicolor characters or bits); and finally, two colors reserved especially for sprites.

Sorry, but we had to be a little more technical this time around. Many of the locations are of value to machine language users; we can't show their features with simple PEEKs and POKEs.

But these locations are powerful, and they are not hard to use once you get a feeling for them.

Next time, we'll take a look at sprites and, literally, fit them into the picture. They are great fun.

Copyright © 1982 Jim Butterfield ©

A Commodore Gotcha

M. G. Ryschkewitsch and M. V. Barnhill

For all Commodore computers, a short hint on how to prevent a hidden error when writing or reading data files to tape or disk.

From the time you first began to learn algebra, you were taught that if X = Y and Y = Z then X = Y. This may seem trivial to you, but the problem is that your PET, VIC, or 64 doesn't always see things this way. However, this state of computer confusion doesn't happen often and is easily taken care of if you know to look for it.

Try typing in and running this disk program (tape users, see below):

```
10 CLOSE 1:OPEN 1,8,15,"SØ:TEST":CLOSE 1
20 OPEN 8,8,8,"Ø:TEST,S,W"
30 X=1/3:Y=.333333333
40 PRINT"{CLEAR}";X;Y;X-Y
50 PRINT#8,X;",";Y;CHR$(13);
60 CLOSE 8:OPEN 8,8,8,"Ø:TEST,S,R"
70 INPUT#8,X,Y:CLOSE 8
80 PRINT X;Y;X-Y
```

If you hadn't thought carefully about what your Commodore computer does with numbers, you were probably surprised that you didn't get the same result both times. What happened? Well, when the computer writes a number to the disk, it sends character by character exactly what it would write to the screen if you asked it to print that number (all of these comments also apply to tape files).

Since the internal operations are carried out with more significant figures than those displayed on the screen or printed to the data file, you can get a truncation error when doing operations with data from a disk file. In cases where the display uses less than the full number of digits because of trailing zeros, there will be no problem (try replacing the 1/3 with 1/2 and .333333333 with .500000000).

You should watch out for this problem if you are comparing numbers written to a data file to numbers kept in memory (for example, if you store temporary results from calculations or try to verify that data has been written properly to a data file). These situations can be handled by comparing STR$(number in memory) to STR$ (number from the disk). This comparison will not give an error if the numbers were the same to start with, but of course there is no guarantee that the numbers were not slightly different. At least your program will not crash unnecessarily.

Tape Version

```
10 OPEN1,1,1,"TEST"
20 X=1/3:Y=.333333333
40 PRINT"{CLEAR}";X;Y;X-Y
50 PRINT#1,X;",";Y;CHR$(13);
60 CLOSE 1
70 PRINT"REWIND TAPE,THEN TYPE:CONT"
80 STOP
90 OPEN 1,1,Ø,"TEST"
100 INPUT#1,X,Y:CLOSE1
110 PRINTX;Y;X-Y
```

©

General-purpose Speedups For Atari BASIC

D K Titchenell

Do you ever need to quickly move character sets around, to achieve fast vertical P/M motion, to instantly clear out players or missiles? These and many other speed-critical problems can be solved with these short, simple subroutines. You need not understand or write machine language to take advantage of its great speed. (Since BASIC itself is written in machine language (ML), you use it all the time without having to be able to explain exactly how PRINT prints.) Here are some efficient solutions to those programming problems where BASIC is just too slow. The example (Program 1) illustrates how to use these subroutines from BASIC.

BASIC is a comfortable language, very friendly, helpful and delightfully forgiving. Eventually, however, the user discovers that there are a few things that can't really be done in BASIC. Then the only solution seems to be to walk over into the rather less certain world of machine language.

There are no two ways about it – BASIC is slow – and you occasionally need to make several things happen with apparent simultaneity.

The Atari provides a very convenient channel for BASIC to communicate with machine language (ML), the USR statement. USR, allowing as it does the passage of any number of values from BASIC to ML, permits a great deal of flexibility. Not only is the number of parameters unlimited, but this number is a known quantity once in the routine and therefore it may be treated as a variable. In taking advantage of this feature, I have found that a relatively small selection of ML routines may be used to solve a large percent of these problems requiring machine language solutions.

MultiPOKE

Under certain circumstances it can be necessary to change the contents of a few addresses without the noticeable lag time between the operations that you encounter using BASIC. One instance of this is in playing music. When BASIC plays a piece with multiple voices, a sharp *attack* (the start of a sound) cannot be achieved because the attack of the different voices is slightly staggered due to the sluggishness of the language.

This point is brought out in *De Re Atari*, and a short machine language routine is presented as a remedy. Another problem of this kind occurs whenever a two-byte register must be changed in realtime – scrolling, for example. Inevitably, that perceptible interval between changing the low byte and the high byte of a register will cause embarrassment. As ANTIC goes zipping through the load memory scan 60 times a second, it can easily display several screens of material during that interval. You could, of course, write little ML routines to solve these individual problems as they arise; or better yet, if you are a little lazy or simply not overly enamored of machine language, you could write a program to solve that type of problem in general.

This was the intention behind MultiPOKE. The MultiPOKE routine acts just like several POKE statements together, performed at machine language speed. Since the number of parameters passed in the USR function is a known quantity, any number of addresses and data to be POKEd into them may be contained in the parameter list. They follow the same order as the POKE statement. The general format is:

D = USR(ADR(POK$),ADDRESS, DATA, [ADDRESS, DATA ...])

A special feature of the routine was added specifically to address the high-byte, low-byte problem. If a data element passed is a one-byte quantity (less than 256), then the routine acts just like one or more POKE statements. If, however, a larger quantity is passed, the low byte is POKEd in in the normal fashion, and the high byte is POKEd into the next higher register in standard low-byte, high-byte form. This eliminates the bother of calculating the carry. Consider the following solution to the scrolling problem:

```
DLIST=PEEK(560)+256*PEEK(561): FOR I=
300 TO 20000 STEP 40: D=USR(ADR(POK$)
,DLIST+4,I): NEXT I
```

This is a very simple way to scroll the screen RAM through most of memory; DLIST + 4 and DLIST + 5 (the LMS operand) are adjusted without BASIC.

Moving RAM With MOV$ And MOVU$

The MOV$ and MOVU$ routines solve a different type of problem: moving large, contiguous areas of RAM. When used in various ways, these utilities can perform the following functions: rapid player/missile vertical motion; initializing areas of memory to a single value or a repeating set of values; or moving around blocks of RAM, such as character sets, with no wasted time. The general form of the call to these routines is: D=USR (ADR(MOV$),FROM,TO,HOWMUCH) where FROM and TO are addresses of origin and destination and HOWMUCH is the number of bytes to move.

The routines are used in exactly the same way, but for complete versatility both are needed. Bytes are moved from the origin to the destination areas one at a time. MOV$ starts at the bottom and goes up. MOVU$ starts at the top and goes down. If the locations of origin and destination do not intersect, both perform identically; if there is overlap, though, the right routine must be chosen for the data to remain intact.

Here's why: suppose you wanted to move five bytes starting at location 500 to the five bytes starting at location 499. MOV$, whose execution proceeds up, would perform correctly – moving the byte at 500 to 499, then the byte at 501 to 500, etc., leaving the data intact. If, on the other hand, you were to use MOVU$, which proceeds down, the following would be the case: the byte at location 504 would be moved to 503, then the byte at 503 would be moved to 502 and so on, filling all five bytes with the original contents of location 504. Both effects can be very useful, but make sure to choose the right one. Let's see some examples.

Speeding Up P/M Graphics With MOV$

Vertical player/missile motion in BASIC tends to resemble an inchworm crawling up the screen. Some alternative methods I have seen have used string manipulation or dedicated machine language programs which erase the former image and position the new one rapidly. Using MOV$ is far simpler than either. You need only to put your player data into a string or other safe place with a zero or two before and after it and "MOV$" that data to the appropriate position in player/missile RAM. No erasing of the former player data is necessary because the incoming data (with help from the zeroes before and after) will obliterate it.

The simple example (Program 1) just puts a player, movable by joystick 1, on the screen, while playing a three-voice melody. It demonstrates the use of POK$ in playing multiple voice music and uses MOV$ for vertical player motion and RAM initialization. The three subroutines at lines 2000, 2100 and 2200 read the machine language code for POK$, MOV$ and MOVU$ into their respective strings.

This is, of course, a terribly inefficient use of time and space, but it is the only method possible when readable, printable characters are required. After entering these routines, you may then convert them into character strings using the following method: call the reading subroutine for POK$, for example, then enter the following line in direct mode:

```
FOR W=1 TO LEN(POK$):?CHR$(27);POK$(W,W);
  :NEXT W
```

This will print out the character string, which then may be made into an assignment statement by putting double quotes at either end and putting "POK$=" in front of it. Each of the three routines fits easily on a single BASIC line.

The short reading routine at line 1000 sets up the two arrays DIRH(15) and DIRV(15) with direction indicators which are selected during execution by using the value returned by the STICK function as a subscript. This is a useful and very time-efficient device.

Clearing P/M RAM With MOV$

The virtual simultaneity afforded by POK$ is not required in the P/M setup procedure at all, but it is used here in line 230 where it serves well to show the format of the routine call. It's also nice to be able to get all of that picky P/M stuff out of the way in one chunk. Line 240 then shows off one of the applications of MOV$, clearing P/M RAM in a split second.

MOV$ executes a data transfer from the bottom up and can thus be used to move blocks down in memory, leaving them intact. Here, however, it is used in the opposite direction, for a purpose. A zero is POKEd into location PMBASE+512. Then 128 bytes are moved up a distance of one byte, starting at that point. Thus the zero value is passed up from each register to the following one, thereby clearing the entire player area.

The actual program loop is a bare skeleton. In line 310 the player data is moved into P/M RAM with MOV$ passing, as parameters, the address of PLAYER$ (FROM), the P/M position (TO) and the number of bytes, ((HOWMUCH) 10 in this case) as the player is eight bytes high and a zero is added at either end.

Since this move does not involve overlapping, either MOV$ or MOVU$ could have been used with identical results; the choice was arbitrary. Lines 320 and 330 read the STICK value and adjust the X and Y coordinates accordingly. Lines 340 and 350 read the tune data and RESTORE when the end is reached. Line 360 uses POK$ to set the player X position and insert the frequency bytes into AUDF1, AUDF2 and AUDF3, the AUDC registers having been initialized by the SOUND statement in line 250. The piece chosen plays here at an appropriately frenetic pace in the absence of a delay loop. The tempo is sufficiently restrained by the snail's pace data reading speed of BASIC. Were we to retard the loop further with added processing, it would probably be advisable to read the tune data into a string first; this would more than double the tempo.

Notes On Structure

A note on the structure of the ML routines themselves: free memory locations that are safe from the meanderings of BASIC or graphics mode changes are often in high demand. In order not to consume the few safe memory areas at the programmer's disposal, each of these routines is relocatable and is placed in a character string.

Most of the space in the routines is used to handle the stack contents properly; the actual loop in each case requires very little space. Were we to POKE all of the parameters into the correct locations beforehand, the size of the routines would be considerably diminished, but the beauty and generality of the parameter list would be lost. Care must be taken in all the routines to pass the correct number of parameters.

Because the address to which execution will return upon completion of the routine is kept in the stack (just below the passed parameters), exactly the right number of bytes must be pulled off the stack or the computer will never find its way home again. In the case of POK$, the number is a variable and the routine keeps track of how many have been pulled; however, there must be an even number, pairs of [address, data]. MOV$ and MOVU$ each must receive exactly the three parameters: FROM, TO and HOWMUCH.

Whenever starting a new project I have taken to entering a listing of these routines into the program at the outset, confident that I will eventually have a need for them. In most cases I do. Possible applications for these ML BASIC helpers are certainly not limited to the ones presented here. New uses suggest themselves often.

Program 1.

```
50 GRAPHICS 0:SETCOLOR 2,3,3
100 GOSUB 2000:REM POK$
110 GOSUB 2100:REM MOV$
120 GOSUB 2200:REM MOVU$
130 GOSUB 2300:REM PLAYER DATA
140 GOSUB 1000:REM DIRECTION ARRAYS
200 TEMP=PEEK(106)-8
210 PMBASE=256*TEMP
220 X=100:Y=50
230 D=USR(ADR(POK$),54279,TEMP,559,46
    ,53277,3,53248,X,704,216):REM ALL
     THOSE PM THINGS
240 POKE PMBASE+512,0:D=USR(ADR(MOV$)
    ,PMBASE+512,PMBASE+513,128):REM C
    LEAR PM RAM QUICKLY
250 SOUND 0,0,10,6:SOUND 1,0,10,6:SOU
    ND 2,0,10,6
260 RESTORE 3100
290 REM ********* LOOP **********
310 D=USR(ADR(MOV$),ADR(PLAYER$),PMBA
    SE+512+Y,10):REM MOVE IN PLAYER
320 TEMP=STICK(0)
330 Y=Y+DIRV(TEMP):X=X+DIRH(TEMP)
340 READ A:READ B:READ C
350 IF C=-1 THEN RESTORE 3101:GOTO 34
    0
360 D=USR(ADR(POK$),53248,X,53760,A,5
    3762,B,53764,C)
400 GOTO 310:REM ****** END LOOP ***
999 REM READ DIRECTION ARRAY
1000 DIM DIRV(15):DIM DIRH(15)
1005 RESTORE 1100
1010 FOR W=5 TO 15
1020 READ Q
1030 DIRH(W)=Q
1040 READ Q
1050 DIRV(W)=Q
1060 NEXT W
1070 RETURN
1100 DATA 1,1,1,-1,1,0,0,0,-1,1,-1,-1
     ,-1,0,0,0,0,1,0,-1,0,0
1999 REM SET UP POK$ ROUTINE
2000 DIM POK$(25):RESTORE 2005
2002 FOR W=1 TO 25:READ P:POK$(W,W)=C
     HR$(P):NEXT W
2003 RETURN
2005 DATA 104,74,170,160,0,104,133,25
     5
2006 DATA 104,133,254,104,240,4,200,1
     45
2007 DATA 254,136,104,143,254,202,208
     ,237
2008 DATA 96
2099 REM SET UP MOV$ ROUTINE
2100 DIM MOV$(39):RESTORE 2105
2102 FOR W=1 TO 39:READ P:MOV$(W,W)=C
     HR$(P):NEXT W
2103 RETURN
2105 DATA 104,104,133,215,104,133,214
     ,104
2106 DATA 133,217,104,133,216,104,133
     ,218
2107 DATA 104,170,160,0,177,214,145,2
     16
2108 DATA 200,208,4,230,215,230,217,2
     02
2109 DATA 208,242,198,218,16,238,96
2199 REM SET UP MOVU$ ROUTINE
2200 DIM MOVU$(47):RESTORE 2205
2202 FOR W=1 TO 47:READ P:MOVU$(W,W)=
     CHR$(P):NEXT W
2203 RETURN
2205 DATA 104,104,133,255,104,133,254
     ,104
2206 DATA 133,253,104,133,252,104,170
     ,24
2207 DATA 101,255,133,255,138,24,101,
     253
2208 DATA 133,253,104,168,177,254,145
     ,252
2209 DATA 136,192,255,208,247,198,253
     ,198
2210 DATA 255,202,224,255,208,238,96
2300 REM SET UP PLAYER DATA
2310 DIM PLAYER$(10):RESTORE 2350
2320 FOR W=1 TO 10
2330 READ P:PLAYER$(W,W)=CHR$(P)
2340 NEXT W:RETURN
2350 DATA 0,255,129,129,129,129,129,1
     29,255,0
```

```
3100 REM MUSIC DATA
3101 DATA 121,0,0,121,0,0,96,128
3102 DATA 0,91,128,0,81,144,0,81
3103 DATA 144,0,121,162,0,121,162,0
3104 DATA 144,182,0,144,182,0,108,182
3105 DATA 0,121,182,0,128,162,0,144
3106 DATA 162,0,128,162,0,162,162,0
3107 DATA 121,243,0,0,243,0,121,243
3108 DATA 0,108,243,0,96,243,0,108
3109 DATA 243,0,96,243,0,121,243,0
3110 DATA 108,162,0,96,162,0,108,144
3111 DATA 0,121,144,0,128,217,0,144
3112 DATA 217,0,128,162,0,162,162,0
3113 DATA 121,121,0,121,121,0,96,128
3114 DATA 0,91,128,0,81,144,0,81
3115 DATA 144,0,121,162,0,121,162,0
3116 DATA 144,182,0,144,182,0,108,182
3117 DATA 0,121,182,0,128,162,0,144
3118 DATA 162,0,128,162,0,162,162,81
3119 DATA 121,243,40,108,243,40,96,24
     3
3120 DATA 45,121,243,47,108,162,53,96
3121 DATA 162,60,108,162,64,121,162,7
     2
3122 DATA 128,217,81,144,217,91,128,1
     62
3123 DATA 96,162,162,108,121,243,121,
     121
3124 DATA 243,121,121,243,121,121,243
     ,121
3126 DATA 0,0,0,0,0,0,0,0,0,-1,-1,-1
```

Program 2.

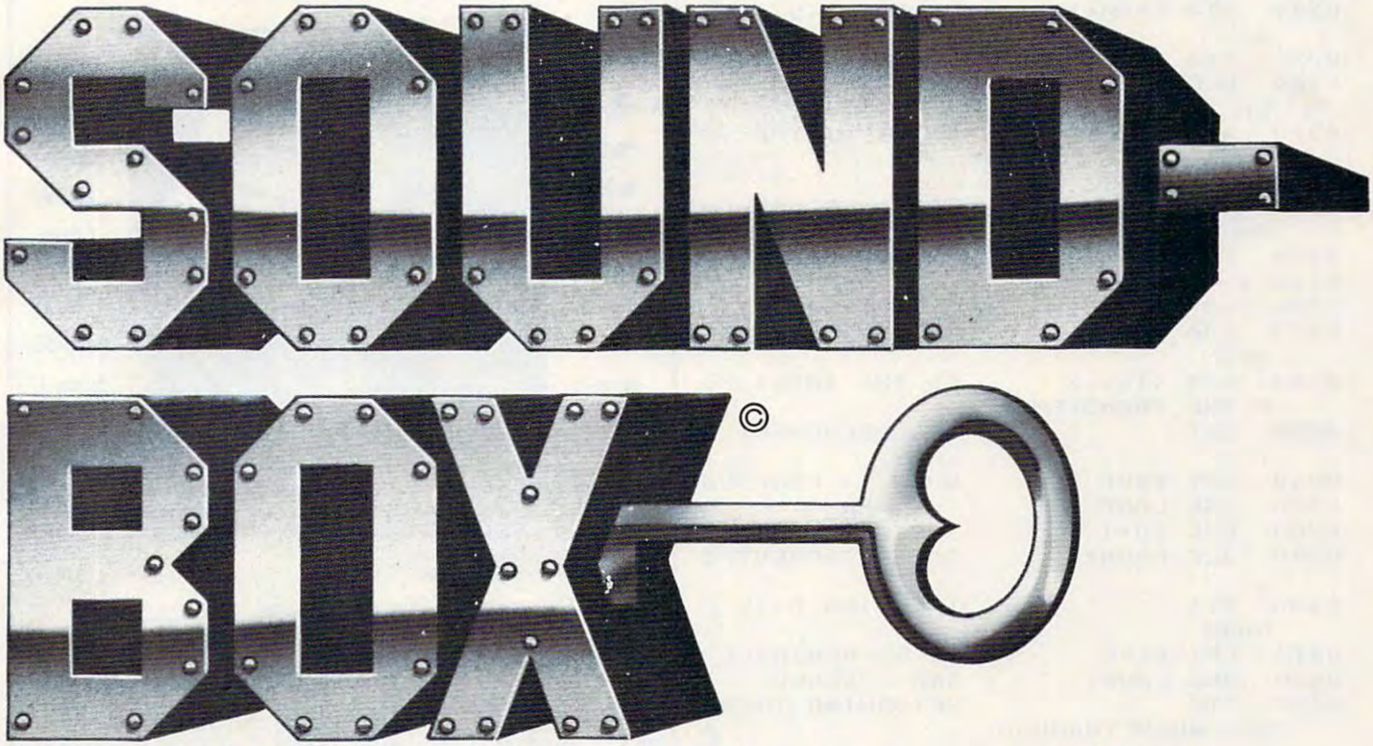
```
10 ; POK$ SUBROUTINE
20 ; BY D. K. TITCHENELL
30 ; PARAMETERS PASSED:
40 ; ANY NUMBER OF PAIRS OF [ADDRESS,
  DATA]
50 ;
60 ;
70 ;
80  *=$0600              ARBITRARY STARTING
  POINT
90 ADDR=$FE             A FREE ZERO PAGE S
  POT
0100 PLA                 NUMBER OF PARAME
  TERS PASSED
0110  LSR A              DIVIDED BY 2
0120  TAX                KEEP IN X
0130  LDY #$00           Y INDEX ZERO
0140 ;                   LOOP FOR EACH PO
  KE
0150 LOOP
0160  PLA                HIGH BYTE OF ADD
  RESS
0170  STA ADDR+1
0180  PLA               LOW BYTE OF ADDRES
  S
0190  STA ADDR
0200  PLA               HIGH BYTE OF NUMBE
  R TO BE POKED
0210  BEQ SKIP          IGNORE IF ZERO
0220  INY               IF NOT ZERO,
0230  STA (ADDR),Y      STORE IN NEXT REG.
0240  DEY               ZERO Y
0250 SKIP PLA           LOW BYTE
0260  STA (ADDR),Y      STORE IN REG.
0270  DEX               COUNT DOWN
0280  BNE LOOP          LOOP IF NOT ZERO
0290  RTS               RETURN TO BASIC
```

Program 3.

```
01 ; MOV$ SUBROUTINE
02 ; BY D. K. TITCHENELL
03 ; PARAMETERS PASSED: FROM, TO, HOW
 MUCH
04 ;
05 ;
10 *=$0600                  ARBITRARY STA
  RTING POINT
20 FROM=$D6                 ZERO PAGE SPO
  TS FOR STORING ADDRESSES
30 TO=$D8
40 LENGTH=$DA               AND LENGTH
50  PLA                     PULL PERAMETE
  RS OFF OF STACK FIRST IS NOT USED
60  PLA
70  STA FROM+1              HIGH BYTE OF
  ORIGIN RAM
80  PLA
90  STA FROM                LOW BYTE
0100  PLA
0110  STA TO+1              HIGH BYTE OF
  DESTINATION RAM
0120  PLA
0130  STA TO                LOW BYTE
0140  PLA
0150  STA LENGTH            HIGH BYTE OF
  LENGTH
0160  PLA
0170  TAX                   LOW BYTE IN X
0180  LDY #$00              ZERO Y
0190 LOOP LDA(FROM),Y       LOAD A BYTE F
  ROM FROM
0200  STA (TO),Y            STORE IN TO
0210  INY                   INCREMENT Y
0220  BNE SKIP              IF <>0 SKIP
0230  INC FROM+1            INCREMENTING
0240  INC TO+1              HIGH BYTES
0250 SKIP DEX               DECREMENT LOW
  BYTE COUNTER
0260  BNE LOOP              LOOP IF<>0
0270  DEC LENGTH            IF ZERO DECRE
  MENT HIGH BYTE
0280  BPL LOOP              RELOOP IF POS
  ITIVE
0290  RTS                   RETURN TO BAS
  IC
```

Program 4.

```
10 ; MOVU$ SUBROUTINE
20 ; BY D. K. TITCHENELL
30 ; PARAMETERS PASSED:
40 ; FROM, TO, HOWMUCH
50 ;
60 ;
70 ;
80 *=$0600                  ARBITRARY START
  ING POINT
90 FROM=$FE                 ZERO PAGE LOCAT
  IONS
0100 TO=$FC
0110  PLA                   FIRST BYTE NOT
  USED
0120  PLA                   HIGH BYTE OF FR
  OM
0130  STA FROM+1            STORE IT
0140  PLA                   LOW BYTE
0150  STA FROM
0160  PLA                   HIGH BYTE OF TO
0170  STA TO+1              STORE IT
0180  PLA                   LOW BYTE
0190  STA TO
0200  PLA                   HIGH BYTE OF HO
  W MUCH
0210  TAX                   IS STORED IN X.
0220  CLC
```



Unlocks your hidden sounds

Now you can access all those sounds you programmed with your Atari 800 in seconds because Soundbox will keep them on file! Soundbox is actually a directory of your sounds. Save laser sounds, doorbells, buzzers, drones, sirens and much more. Simply name your programs and store them. Later, "page" through Soundbox and pick a sound, RUN it and LIST to screen, if you like. It's easy for first-time Atari users as well as experts. Soundbox disks hold about 100 programs, four on a page. The expansion disk also holds 100. Why search for hours to find that perfect sound … when it's right in your Soundbox!

Martin Acceleration, Ltd., Wilmington, Delaware Dealer Inquiries Invited

Inquire at your local store or Mail Order Form to: Soundbox Division, c/o Martin Acceleration, Ltd., P.O. Box 1323, Bethlehem, PA 18016

Order Form

Inquire at your local store or Mail Order Form to: Soundbox Division,
c/o Martin Acceleration,
Ltd., P.O. Box 1323,
Bethlehem, PA 18016

TM Atari Corporation
Soundbox was written by
Martin K. Romanchuk

Please send me the following Soundbox disks.

______ Soundbox for Atari^{TM} 48K (only) at $39.95 each

______ With expansion disk at $49.95 per set

______ Expansion disk separately at $19.95 each

Name______________________________

Address______________________________

City______________________________ State______ Zip______

Telephone______________________________

____ A certified check or money order is enclosed in the amount of __________.
PA residents add 6% sales tax.

____ Please charge my __ Visa __ MasterCard

Card Number______________________ Expiration Date__________

Signature______________________________
(required for credit card purchases)

Free Soundbox t-shirt while supplies last. State size S____ M____ L____

```
0230   ADC FROM+1          HIGH BYTE OF
0240   STA FROM+1          HOWMUCH IS ADDE
    D
0250   TXA                 TO FROM AND TO
0260   CLC                 AS EXECUTION WI
    LL
0270   ADC TO+1            START AT THE TO
    P
0280   STA TO+1
0290   PLA                 LOW BYTE OF HOW
    MUCH
0300   TAY                 IS KEPT IN Y
0310 ;
0320 LOOP
0330   LDA (FROM),Y        WHICH IS THEN U
    SED
0340   STA (TO),Y          AS THE INDEX FO
    R THE TRANSFER
0350   DEY                 AND DECREMENTED
    .
0360   CPY #$FF            WHEN IT REACHES
0370   BNE LOOP            <0 THEN
0380   DEC TO+1            THE HIGH BYTES
0390   DEC FROM+1          ARE DECREMENTED
    .
0400   DEX                 THE HIGH BYTE C
    OUNT
0410   CPX #$FF            IS DECREMENTED
0420   BNE LOOP            AND CHECKED
0430   RTS                 RETURNING TO BA
    SIC WHEN THROUGH
```

©

BASIC Trace For The VIC

Jim Wilcox

A trace *program is very helpful as a tool for finding errors, for* debugging. *It will let the operator see at which line number the program is,* while it is running. *The programmer can see errors in program flow as well as find the areas where problems are located.*

This "Trace" program will adjust to any size memory in the VIC. It uses fewer than 200 bytes of memory. The program will be protected in the uppermost memory positions.

Type in the BASIC program below and double check the DATA statements. Once you're sure the program is right, SAVE it. RUN the program, and in a couple of seconds the screen should show you the SYS values to turn the TRACE on and off. On an unexpanded VIC the following will be printed:

```
TRACE ON  SYS( 7501 )
TRACE OFF SYS( 7488 )
```

If, when you type SYS (7501) on an unexpanded VIC, the machine "crashes," or does something unexpected, something was typed in wrong. The solution is to LOAD the program and correct any mistakes, then try again.

After you've got a working tracer, the program that you would like TRACEd may now be LOADed or typed in. Type in the SYS and hit RETURN to turn the TRACE on and RUN the program. The line numbers will be printed in reverse field.

Trace will help you find bugs faster and correct them. It will also show exactly where a program is operating at any given time during execution.

```
5 F=Ø:C=PEEK(55)-192:IFC<ØTHENC=C+256:F=
   -1
1Ø D=PEEK(56)+F:POKE55,C:POKE56,D:CLR
15 N=PEEK(55)+256*PEEK(56)
2Ø F=Ø:FORD=NTON+191:READA$:IFASC(A$)<58T
    HENA=VAL(A$):GOTO35
25 IFASC(A$)=76THENA=VAL(RIGHT$(A$,LEN(A$
    )-1))+PEEK(55):IFA>255THENA=A-256
    :F=1
3Ø IFASC(A$)=72THENA=VAL(RIGHT$(A$,LEN(A$
    )-1))+PEEK(56)+F:F=Ø
35 POKED,A:NEXT
4Ø PRINT"TRACE ON  SYS("N+13")"
45 PRINT"TRACE OFF SYS("N")":NEW
5Ø DATA169,23Ø,133,115,169,122,133,116,16
    9,2Ø8,133,117,96,169,255,141,61,3
    ,169,76
55 DATA133,115,169,L31,133,116,169,HØ,133
    ,117,96,72,138,72,152,72,165,58,2
    Ø1,25Ø
6Ø DATA176,12,2Ø5,61,3,2Ø8,1Ø,165,57,2Ø5,
    6Ø,3,2Ø8,3,76,L134,HØ,165,57,141
65 DATA6Ø,3,141,62,3,165,58,141,61,3,141,
    63,3,169,18,32,21Ø,255,169,32
7Ø DATA32,21Ø,255,169,Ø,141,64,3,162,Ø,32
    ,L148,HØ,173,65,3,24Ø,3,238,64
75 DATA3,173,64,3,24Ø,8,173,65,3,9,48,32,
    21Ø,255,232,224,5,2Ø8,227,173
8Ø DATA64,3,2Ø8,5,169,48,32,21Ø,255,169,1
    46,32,21Ø,255,1Ø4,168,1Ø4,17Ø,1Ø4
    ,23Ø
85 DATA122,2Ø8,2,23Ø,123,76,121,Ø,169,Ø,1
    41,65,3,56,173,62,3,253,L182,HØ
9Ø DATA168,173,63,3,253,L187,HØ,144,12,23
    8,65,3,141,63,3,14Ø,62,3,76,L153
95 DATAHØ,96,16,232,1ØØ,1Ø,1,39,3,Ø,Ø,Ø,3
    2,56,53,32,4,1,2Ø,1
```

©

ATARI

ComputAbility™ presents the Newest in Software Products for Atari

SUPER SPECIALS

CHOPLIFTER
$22.95 DISK

GENETIC DRIFT
$20.49 DISK/TAPE

SHOOTING ARCADE
$20.49 DISK/TAPE

K-RAZY SHOOT OUT
$31.95 CARTRIDGE

STRATOS
$24.49 DISK/TAPE

BAJA BUGGIES
$22.49 DISK/TAPE

Prices effective March 1 through March 31, 1983

FREE* SOFTWARE

FREE* SLIK STIK

ATARI	
CONVERSATIONAL LANGUAGES - T	43.95
INVITATION TO PROGRAMMING 2 & 3 - T	21.95
MUSIC COMPOSER - C	32.95
MY FIRST ALPHABET - D	26.95
TOUCH TYPING - T	19.95
HOME FILING MANAGER-D	37.95
MAILING LIST - T	19.95
ASTEROIDS - C	26.95
CAVERNS OF MARS - D	28.95
COMPUTER CHESS - C	26.95
MISSILE COMMAND - C	26.95
SUPER BREAKOUT - C	26.95
STAR RAIDERS - C	32.95
ASSEMBLY EDITOR - C	44.95
BASIC - C	44.95
MACROASSEMBLER - D	65.95
MICROSOFT BASIC - D	65.95
PILOT (HOME PACKAGE)-C	58.95
INVITATION TO PROGRAMMING I - T	18.95
SPEED READING - T	55.95
BASKETBALL - C	26.95
GRAPH-IT - T	15.95
JUGGLE'S HOUSE - D/T	22.95
PILOT (EDUCATOR) - C	97.95
VIDEO EASEL - C	26.95
DEFENDER - C	32.95
GALAXIAN - C	32.95
QIX - C	32.95
DIG DUG - C	32.95
ET - C	38.95
TIME WISE - D	23.95
ATARI WRITER - C	61.95

ADVENTURE INTERNATIONAL	
PREPPIE - D/T	23.95
S.A.G.A. ADVENTURES - D	31.95
SEA DRAGON - D/T	27.95
STRATOS - D/T	27.95
BUG OFF - D/T	23.95
ANALOG	
RACE IN SPACE - D/T	20.95
CARNIVAL - D/T	20.95
SUNDAY DRIVER - D/T	23.95
CRASH DIVE! - D/T	23.95

ON-LINE	
JAWBREAKER - D/T	23.95
ULTIMA I - D	31.95
THRESHOLD - D	31.95
ULTIMA II - D	44.95
MOUSKATTACK - D	27.95
FROGGER - D/T	27.95
SIRIUS	
BANDITS - D	27.95
WAY OUT - D	31.95
BEER RUN - D	23.95

THE DISCOUNT SOFTWARE COMPANY THAT PAYS YOU A DIVIDEND!!!

DATASOFT	
SHOOTING ARCADE - D/T	23.95
PACIFIC COAST HIGHWAY - D/T	23.95
MICROPAINTER - D	27.95
CANYON CLIMBER - D/T	23.95
FATHOMS FORTY - D	27.95
O'RILEY'S MINE - D/T	27.95
ROSEN'S BRIGADE - D/T	27.95
SANDS OF EGYPT - D	31.95
BIG FIVE	
COAL MINER 2049'ER - Cart	39.95

I.D.S.I.	
POOL 1.5 - D	27.95
POOL 400 - Cart	31.95
SPEEDWAY BLAST - Cart	31.95
JUGGLER - D	23.95
SURVIVAL OF THE FITTEST - Cart	31.95
FIRST STAR	
ASTRO CHASE - D/T	23.95
SENTIENT	
CYBORG - D	27.95
GOLD RUSH - D/T	27.95

WE CARRY HUNDREDS OF ITEMS FOR ATARI 400/800, ASK FOR OUR FREE CATALOG.

*COUPON PROGRAM

The purchase of each program (with the exception of Super Specials, Atari, and APX) will earn you 1 COMPUTABILITY DIVIDEND COUPON. Save 3 coupons and redeem them for your choice of Slik Stik, a Lefty Adaptor, or an Extension Cable. OR save 10 coupons and redeem them for your choice of any program we sell for $24.00 or less (with the exception of Super Specials, Atari, and APX). You pay only a $2.50 shipping and handling charge.

BRODERBUND	
APPLE PANIC - D/T	23.95
STELLAR SHUTTLE - D/T	23.95
DAVID'S MIDNIGHT MAGIC - D	27.95
STAR BLAZER - D	25.50
TRACK ATTACK - D	23.95
LABYRINTH - D/T	23.95
SERPENTINE - D	27.95
DUELING DIGITS - D	23.95
DEADLY SECRETS - D	27.95
CHOPLIFTER - D	27.95
GENETIC DRIFT - D/T	23.95
SPINNAKER	
SNOOPER TROOPS #1 - D	35.95
SNOOPER TROOPS #2 - D	35.95
FACEMAKER - D	27.95
STORY MACHINE - D	27.95

EDU-FUN
CALL FOR ITEMS AND PRICES

THORN
CALL FOR ITEMS AND PRICES

NEW ITEMS	
PIG PEN - D	23.95
BAJA BUGGIES - D/T	25.50
STARBOWL FOOTBALL - D/T	25.50
MASTER TYPE - D	31.95
ALI BABA - D	26.50
JEEPERS CREEPERS - D	23.95
PAINT - D	33.95
KID GRID - D/T	23.95
MOSAIC 32K	**97.95**

D - Disk T - Cassette
C - Cartridge

ATARI is a trademark of ATARI, Inc.

Starfighter

The Ultimate Joystick
- 2 Year Warranty
- More Accurate
- Easier to Hold

$16.95

SLIK STIK

- 90 Day Warranty
- Easy Ball Top Control

Suncom

$9.95

LEFTY JOYSTICK ADAPTOR
Adapts to *any* Atari Joystick.
Moves fire button to top Right. $9.95

EXTENSION CABLE (5 ft.)
Adapts to *any* Atari controller. $6.95

ALL JOYSTICKS WORK WITH Atari VCS, Sears Telegame, Commodore VIC 20, Atari 400/800 (All products have registered trademarks)

30 Day Money Back Guarantee on all Suncom Products - Dealers inquiries invited!!!

Mastercard/VISA
Order Toll Free

MasterCard

800-558-0003
No surcharge for credit cards

VISA

In Wisc. Call
414/351-2007

ORDERING INFORMATION

To order by mail send money order, certified check or personal check (allow 14 days to clear) to COMPUTABILITY. Include $2.00 shipping on software orders and $2.50 shipping on hardware orders (FREE OR PURCHASED). Mastercard & VISA please include card number and expiration date. WI residents please add 5% sales tax. Outside of continental U.S.A. please add 15% shipping (U.S. Funds only). Prices subject to change without notice.

Order Hours:
Mon. - Fri. 12 pm - 9 pm C.S.T.
Sat. 12 pm - 5 pm C.S.T.

ComputAbility
P.O. Box 17882
Milwaukee, WI 53217

Atari Mass Line Delete

Aaron M. Contorer

Probably just about everyone who has tried to write a lengthy BASIC program on the Atari has encountered the most serious bug in Atari BASIC: system lockup. When changes are made in a program so that a major portion of the program must be moved either up or down in memory, BASIC can cause the computer to completely stop working, or to "lock up." In such cases, there is usually no recourse but to turn the computer off and back on, completely erasing the memory. One good solution is this program, "Mass Line Delete."

If changes in a BASIC program are made by the program itself (through the use of the "dynamic keyboard"), system lockup will not occur. "Mass Line Delete" will erase a specified area of a target program – for example, lines 100 through 200 – without causing the computer to crash. Use of Mass Line Delete also saves a considerable amount of typing, since the only line numbers the user must type are the start and end of the area to be erased, as opposed to typing in the number of every line in the area.

The program itself is as brief as possible so that it will leave most of the computer's memory for the main program. To use Mass Line Delete, first type it in, then list it to cassette by typing LIST "C" or to disk by typing LIST "D:DELETE. LST". Then, at any point during work on another program that you wish to use Mass Line Delete, retrieve it using the ENTER command and type GOTO 32600. In response to the utility's prompt, type the starting and ending line numbers of the program area to be deleted. Mass Line Delete will delete the appropriate lines and automatically end. You may then continue work on your program.

Remember that the safest way to make sure that your program is not lost forever is to SAVE it at least every half-hour that you work on it.

```
32600 GRAPHICS 0:? :? "Mass Line Dele
      te"
32605 TRAP 32605:POKE 84,11:? "Start,
       end";:INPUT S,E
32610 IF INT(S)<>ABS(S) OR S>32099 OR
       INT(E)<>ABS(E) OR E>32099 OR E
      <S THEN ? CHR$(253):GOTO 32605
32615 GRAPHICS 0:? :?
32620 ? S:S=S+1
32625 ? "CONT":POSITION 0,0:POKE 842,
      13:STOP
32630 POKE 842,12:IF S<=E THEN 32615
32635 GRAPHICS 0:END
```

©

The Confusing Catalog

Jim Butterfield, Associate Editor

Have you ever wanted to have a program gain control of the disk catalog? There are a number of ways to use directory information, but getting hold of it is not as simple as it might seem at first glance.

On 4.0 Commodore machines, you just type CATALOG or DIRECTORY. On earlier machines, you must LOAD "$",8 and then LIST. Either way, you get a directory with your disk header, information on the programs, and the number of bytes free. Very handy indeed.

Here's the problem: you would like your program to be able to read a directory. It seems simple: just OPEN it as a file and bring in the items. Unfortunately, it doesn't work that way.

Two Types

When you command LOAD "$",8 you are bringing in a directory with a LOAD command; it arrives in a certain format. If you OPEN 1,8,2,"$" within your program, you'll get an entirely different format. Why?

When you say LOAD, the disk manufactures a directory that imitates a BASIC program. After all, the next thing you'll say is LIST, and the only thing that can be listed is BASIC. If you say OPEN, however, the disk will give you its directory, in binary, just as it is stored on the disk surface. That seems to be a little better – until you realize that BASIC has a devil of a time understanding binary.

You can do an OPEN and get the "imitation program." The trick is to use secondary address 0 – usually reserved for LOADing.

Another Problem

Either way, you get binary. You'll need to translate it and interpret it; and you'll need to cope with that annoying BASIC glitch, inputting a CHR$(0). Whenever BASIC GETs a CHR$(0), it changes it to a null string (""), and you'll need to detect this and change it back.

The coding for this is fairly easy. After we get a character with GET A$, we may take its binary value with A=ASC(A$) – except that the null string won't work right. So, we say, A=ASC (A$+CHR$(0)) and everything works out.

Imitation BASIC

This is the easiest and most standard way of obtaining directory information; it works the same way with all Commodore disks. To understand it, we must see how a BASIC line is constructed:

First two bytes:	forward chain or zero (dummy on directory)
Next two bytes:	binary number
Then:	text of line
Ending with:	binary zero

So let's write it:

```
100 OPEN 1,8,0,"$0"
      Let's get the directory for drive 0.
110 N$=CHR$(0)
      Here's our null string replacement.
190 GET#1,A$,A$
      Skip the "Load Address" at file start.
200 GET#1,A$,A$
      Skip the forward chain, except:
210 IF A$="" GOTO 400
      Zero chain means the end.
220 GET#1,A$,B$
      Get the binary number.
230 PRINT ASC(A$+N$)+ASC(B$+N$)*256;
      Print "number of blocks."
300 GET#1,A$
      Let's get text.
310 IF A$="" THEN PRINT:GOTO 200
      End of this line: go back;
320 PRINT A$;
      Print one character;
330 GOTO 300
      Get some more.
```

This program prints the directory. Big deal: you could do that anyway. But since it's a program, you can change it to do whatever functions you need. For example, you could dig into the text part in more detail, extracting the program name and type; that way, your program would know if a given data file were on the disk.

It's handy to be able to check how many blocks are free on the disk. Our program already does this: the last number that line 230 calculates will be the blocks-free value. You can abbreviate this procedure by making the program skip all the file names. Change the OPEN statement to read:

```
100 OPEN 1,8,0,"$0:S%Q"
```

Now, the program will catalog only those programs whose name happens to be exactly S%Q. Chances are you won't have many of these. Your directory is now shortened down to the header line and the BLOCKS FREE line. Let's telescope our program into a simple block-free checker:

Software for the VIC-20 from
Creative Software

CARTRIDGE GAMES

CHOPLIFTER
This smash hit BRODERBUND game is now available for the VIC-20 in cartridge form. $44.95

APPLE PANIC
A faithful adaptation of the popular Apple Monster game on a VIC-20 cartridge. $44.95

ASTROBLITZ
Given the highest accolades in a popular computer magazine, this TOM GRINER creation offers a fast moving challenge. $44.95

SERPENTINE
Fine translation of the popular game to a VIC-20 cartridge. $44.95

TERRAGUARD
Classic space action game by TOM GRINER. Protect your planet from galactic attackers. $44.95

VIDEOMANIA
TOM GRINER'S game requires you to be aggressive at all times. No rest for the weary in this VIC-20 cartridge. $44.95

TRASHMAN
This game received great reviews in a recent magazine article. You are a spirited citizen who drives a garbage truck and collects debris. $44.95

BLACK HOLE
This game is for adventurous space travelers who understand the fatal attraction of a black hole. $44.95

Software from
UMI

CARTRIDGE GAMES

SPIDERS OF MARS
Author PETER FOKOS has designed one of the smoothest VIC-20 cartridge games. Defend the Martian surface. $40.00

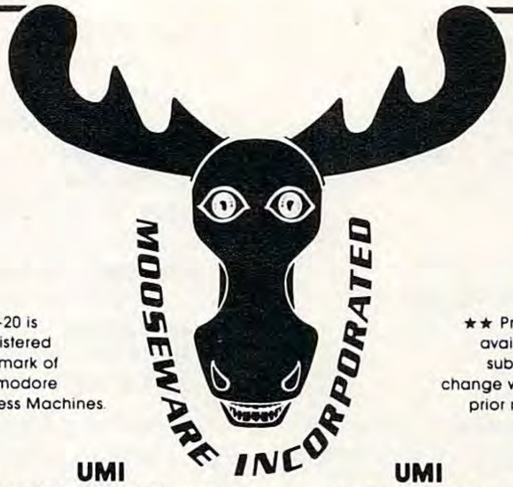
OUTWORLD
TOM GIGUERE has created one of the most visually stunning space games for the VIC-20 cartridge format. $40.00

CLOUDBURST
This game has been playtested by several people in our office and was a resounding success. Another triumph for PETER FOKOS in a VIC-20 cartridge. $32.00

SKIBBEREEN
This is a colorful breakout game by JOANNE LEE. Beware the Loch Ness monsters and get to the pot of gold. $32.00

ALIEN BLITZ
This VIC-20 cartridge was created by the dynamic duo of FOKOS & GIGUERE. It is reminiscent of an all time classic space game. $32.00

RENNAISSANCE
A challenging strategy game based on Reversi. Rennaissance will challenge the analytical VIC-20 user. $40.00



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

★★ Prices & availability subject to change without prior notice.

UMI
CARTRIDGE GAMES cont.

SUPER AMOK
This highly advanced version of the original AMOK game from UMI provides nail bitting suspense! $40.00

METEOR RUN
Traversing the meteor fields on the way to the planet Aldebaran, you must fend off the natural and unnatural threats to your survival. $40.00

ARACHNOID
The most innovative cartridge game we have ever seen. Change several parameters simultaneously to create a tremendous variety of playing conditions. $64.95

SATELLITES & METEORITES
Blast meteorites from outerspace and beware the killer satellites. $40.00

UMI
CASSETTE GAMES

AMOK
Avoid being zapped by the robots of AMOK, as you roam the rooms. $20.00

SUBCHASE
Patrol the seas and rid them of enemy submarines. $20.00

ALIEN BLITZ
The challenge of Alien Blitz will be enjoyed a long time. $20.00

THE ALIEN
The roles reversed, now your the alien wandering the passages of Critter City. $20.00

3-D MAZE
The passageways open realistically as you walk in search of escape. $12.00

RACEWAY
Time, speed, and driving skill are your measure in Raceway. $12.00

KOSMIC KAMIKAZE
Being a pilot of the honorable Kamikaze Space Squadron, you learn to face death with a smile. $20.00

GRANDMASTER
Try to match strategy and skill with a Grandmaster Chess Champion. $32.00

METEOR SHOWER
Avoid the Meteor Storm to survive. $12.00

UMI
BUSINESS SOFTWARE

VICALC
A handy calculator program for your Commodore VIC-20. Cassette $12.00

VITERM B
Use your VIC modem to communicate over the telephone with data bases, mail networks, the SOURCE and larger computers. Cartridge $40.00

VIDATA
This tape is a useful addition to any VIC owner's library. The user can manage inventories or any type of list. Cassette $24.00

VICHECK
A checkbook balancing program on tape. Everyone we know could use this. Cassette $20.00

WORDCRAFT
Bring professional work processing home to VIC. The most complete package available. $269.95

FORTH-20
A high level language, with the power of a low-level language. $79.95

UMI VIC-20
EDUCATIONAL SOFTWARE

SPACEDIV
Engrossing math education cassette that holds a child's interest. Colorful graphics are a big plus. Cassette $12.00

SKYMATH
Good graphics and solid design make this a useful educational tool for teaching addition and subtraction. Cassette $20.00

SUPER HANGMAN
A classic game with educational value. Increases spelling ability in some children. Cassette $16.00

UMI
VIC-20 HARDWARE

VICEPS
At last, a low cost way to connect your VIC-20 to an EPSON MX-80 or MC-100. If your EPSON printer has GRAFTRAX, all the standard VIC-20 characters may be printed. $49.95

BUTI
The acronym stands for "Basic Programmer's Utility ROM". It is a beauty of a product that adds 3K of RAM and seventeen new instructions to the VIC's BASIC language. $69.95

Commodore-64
SOFTWARE

At last, UMI has come to the rescue of the thousands of Commodore-64 owners who are frustrated by the lack of software. The following cassette games require a datassette.

MOTOR MANIA
This road racing game provides a challenging race course for you, the driver. It is one of the first games to show what the 64 can really do. Cassette $29.95

RENNAISSANCE
For the strategists among you, UMI has adapted REVERSI, a classic board game, to the Commodore-64. Cassette $29.95

VICHECK
For those who want to balance their checkbooks quickly and accurately, this is the program for you. Cassette $29.95

MOOSEWARE

THE SPOKESMAN
Mooseware is proud to be the exclusive distributor of the SPOKESMAN, a versatile speech synthesizer for the VIC-20 and the Commodore-64. With its built in vocabulary and self contained memory, very little effort is required to make your programs talk back to you. This synthesizer outperforms units costing much more. $150.00

MOOSE CLUB
So you want to be a MOOSE or what? By sending us a $20.00 initiation fee (thereafter $5.00 annual dues) you will receive a discount of 10% off any software products we sell. This offer is valid only for MOOSE CLUB members in good standing.

TO ORDER SEND CHECK OR MONEY ORDER TO:
MOOSEWARE INCORPORATED
Post Office Box 17868, Irvine, California 92713
— California Residents Add 6% Sales Tax —
All Orders Add $2.00 Shipping

NAME ____________________

ADDRESS ____________________

CITY ____________ STATE ________ ZIP ________

WRITE FOR FREE BROCHURE & ASK ABOUT THE MOOSE CLUB!!!

```
100 OPEN 1,8,0,"$0:E7!N"
      Another unlikely name
110 N$=CHR$(0)
200 GET#1,A$,A$,A$,A$,A$,A$,A$
      Throw away load address, link, number.
210 GET#1,A$:IF A$<>" "GOTO 210
      Throw away the header line
220 GET#1,A$,A$,A$,B$
      Throw away the link, get the number.
230 F=ASC(A$+N$)+ASC(B$+N$)*256
      Here's our block-free count.
400 CLOSE 1
410 PRINT F
```

We've only scratched the surface. Try your hand at programming some directory search function of your choice.

Bit-image Directories

You can get more information from a bit-image directory than from a BASIC-imitator. For example, you can read the length parameter of relative files, see deleted files, and view file track and sector values.

But this comes with considerable difficulty. You might get any one of several different formats, depending on the disk. We won't do the whole job here: you can chase after some of the details for yourself.

```
100 OPEN 1,8,15,"I0":CLOSE 1
      We must initialize for this one.
110 OPEN 1,8,2,"$0"
      Here comes the bit directory.
120 N$=CHR$(0)
130 GET#1,A$
      The disk will identify itself.
140 A=ASC(A$+N$)
      Here's the identity.
150 IF A=67 THEN PRINT "†8050 l"
160 IF A=65 THEN PRINT "†4040 l"
170 IF A=1 THEN PRINT "†2040 l"
      Just to prove we identified it.
      8250's will give trouble here.
200 FOR J=1 TO 253
210 GET #1,A$
220 NEXT J
      Skip the (bit) BAM.
230 IF A<>67 GOTO 300
240 FOR J=1 to 254*2
250 GET#1,A$
260 NEXT J
      The 8050 has a big BAM to skip.
300 FOR J=1 TO 8
      Eight files per block.
310 GET#1,F$,T$,S$
      File type, Track, Sector.
320 F=ASC(F$+N$)
330 P$=" ":FOR K=1 TO 16
      Get 16-character name.
340 GET#1,X$:P$=P$+X$
350 NEXT K
360 FOR K=1 TO 9
370 GET#1,X$
380 NEXT K
      There's useful stuff here; we'll skip it.
390 GET#1,L1$,L2$
      File length
400 IF J<8 THEN GET #1,X$,X$
      Weird; 254 bytes/8 leaves us two bytes short.
410 SW=ST
      To allow us to test end-of-directory.
420 IF F<129 OR F>132 GOTO 480
      Not a real file.
430 PRINT P$;ASC(L1$+N$)+ASC(L2$+N$)*256
      Name and length.
480 NEXT J
500 IF SW=0 GOTO 300
900 CLOSE 1
```

This isn't a program – it's a research outline. Yes, you can go in there and drag out the BAM. Yes, you can dig useful data out of the stuff we skipped in lines 360-380. Check your disk manual for details.

It's not easy either way. The "imitation BASIC" is the shortest and works on all disks: use it when you can. But if you need the extra power of the bit map, don't hesitate to go for it. ©

VERSACALC
VERSACALC
VERSACALC
VERSACALC
VERSACALC
VERSACALC
VERSACALC
VERSACALC
VERSACALC
VERSACALC

**TUTORIAL
UTILITIES
AND FILE MANAGER**

**Everything you always wanted to do* with Visicalc,
(but thought you couldn't).**

YES! We said "SORT VISICALC"!

NOW YOU CAN:

- * SORT a Visicalc screen on any column, ascending or descending; all related formulas and labels are sorted too.
- * put the entire disk CATALOG on the screen at once!
- * easily do Year-To-Date accumulations!
- * "pound" formulas to expose the full formulas in place on the screen![1]
- * append two Visicalc files![1]
- * print the contents of a /SS file!
- * print the contents of a /PF file!
- * AND our EASEL BINDER is so nice that you will put your other manual in it!

Apple II*	$100
PET & CBM*[1]	125
IBM PC[1]	150
Apple III[1]	150

*specify DOS

If you use Visicalc™ but you are bumping into its limitations, then you need Versacalc™! Versacalc runs within Visicalc but uses no extra memory; in fact, it effectively increases memory by letting you call in modules from disk as needed.

A Tutorial section makes clear such features as @LOOKUP, DIF, @NA, @ERROR, which are not well explained in the Visicalc manual.

A Utilities section makes it easy to create your own menu-driven modules which condense hundreds of commands into four keystrokes. You can build in sophisticated error checking (e.g. Is the input value between certain limits?). Now it is possible for people untrained in Visicalc to perform the weekly updating without constant instruction.

Anthro-Digital Software
P.O. Box 1385
Pittsfield, MA 01202
413-448-8278

Apple II is a trademark of Apple Computer, Inc
Versacalc is a trademark of Versacalc Enterprises, Inc.
Visicalc is a trademark of Visicorp Inc

BATTERIES INCLUDED

village by the grange, 71 mccaul st. (f6) toronto m5t 2x1 telephone 596-1405

ARBITER 1.4 MULTI-USER DISK SYSTEM FOR COMMODORE 4.0 COMPUTERS

OVER THREE HUNDRED IN USE ACROSS ONTARIO

Since September 1981 **BATTERIES INCLUDED** has been installing the ARBITER system in classrooms of Commodore BASIC 4.0 computers. The computers are connected to CBM Disk Drives and printers. All users have access to all disk drives and printers plus a host of commands to make this system configuration really usable!

THE ARBITER 1.4 SYSTEM IS READY TO GO!

FEATURES

1) Easy installation.
2) Uses no RAM or Utility Sockets.
3) Up to 32 computers in one system.
4) System self initializes on power up.
5) Operation is completely transparent to the user.
6) Extended commands allow a friendly multi-user environment.
7) System design virtually eliminates interleaved printer output.

$150.00 per unit

SPECIAL COMMANDS

@S– Allows students to protect files with a five character password. A three character user ID is forced into the file name.

@L– Allows the students to load protected files if the password code is known.

LISTC – Used to produce program listings with a Commodore printer. Clumsy OPEN, CMD, LIST, PRINT#, CLOSE sequence not needed. It overcomes the listing problems found on other multi-user hardware systems.

LISTP – Used to get program listings on systems which have an ASCII printer. The cursor control characters are expanded and displayed in brackets. e.g. ‹home›

ALL FILE TYPES ARE SUPPORTED – During relative or sequential file access a delay has been built in so the computer will retain control of the system until the file is closed.

TEACHER UTILITY – A utility is supplied on disk to allow the teacher to produce a hardcopy listing and output from any of the protected or unprotected files selected. Once the files are chosen from the disk directory the teacher may do other tasks while the job is completed.

IF YOUR CLASSROOM WAS DESIGNED TO TEACH COMPUTER LITERACY OR
STRUCTURED BASIC THEN THIS SYSTEM WAS DESIGNED FOR YOU.
Arbiter and Arbiter 1.4 are copyrights of Batteries Included.

Bill Wilkinson

Almost BASIC

This month we'll start a major project: a pseudo-BASIC interpreter written in Atari BASIC. Will this be a useful product? No. First, since it is written in and interpreted by Atari BASIC, it will of necessity be much slower than even Atari BASIC. Second, it will be an extremely limited language (as we'll shortly see) and, in fact, a nonstandard language.

But suppose we could overcome the first objection (speed) and ignore the second (so what if it is nonstandard, as long as it is ours). Would it be useful then? Sure. In fact, we could even speculate on rewriting the interpreter in C/65 or assembly language and ending up with an extremely fast, presumably integer-only interpreter. Still, the language is limited, and it would have to have some major extensions added before it would be really usable.

Enough speculation. Let's proceed to the language's definition.

1. The program editing scheme used will be essentially identical to that of Atari BASIC. Line numbers from 1 to some maximum will automatically be sorted and executed in order. Entering just a line number will erase any line with that number.

2. Single letter variables *only* will be allowed. This is a major point of departure from Atari BASIC, but it makes the interpreter significantly simpler. And no string variables.

3. Only the first letter of each statement name (command name) will be significant. Another big departure, and one which limits us to 26 different statements. Also note that this implies that if we use "Print," we can't use "Plot," "POKE," or "Position," etc. This also implies that you can keep programs small (and unreadable) by using single letter commands.

4. No functions. Sorry, but there will be no "RND(0)", no "SIN(30)", etc. This is necessary if we are to keep the expression analyzer down to manageable proportions when it is written in Atari BASIC.

5. No precedence of operators. Same excuse as number 4. This means that "3+4*5" will evaluate as "(3+4)*5" or 35. Most BASICs would see that as "3+(4*5)" or 60. Similarly, no parentheses will be allowed.

6. No provision for loading or saving programs. It would be easy to add this, and we might do so later. However, I see little point in doing so as long as the interpreter is running under Atari BASIC.

Whew! Feel restricted? Well, if you are adventuresome, you can try adding to and modifying the interpreter. It is a good exercise in logic, and you might even get good enough at it to give us a scare.

And one more thing before we get started with the heavy stuff. What do we call this thing? I haven't come up with anything better than BAIT, which is my acronym for BASIC (Almost) InTerpreter. (And which is also meant to imply that it *is* bait: I am fishing for innovation and interest from you, my gentle readers.)

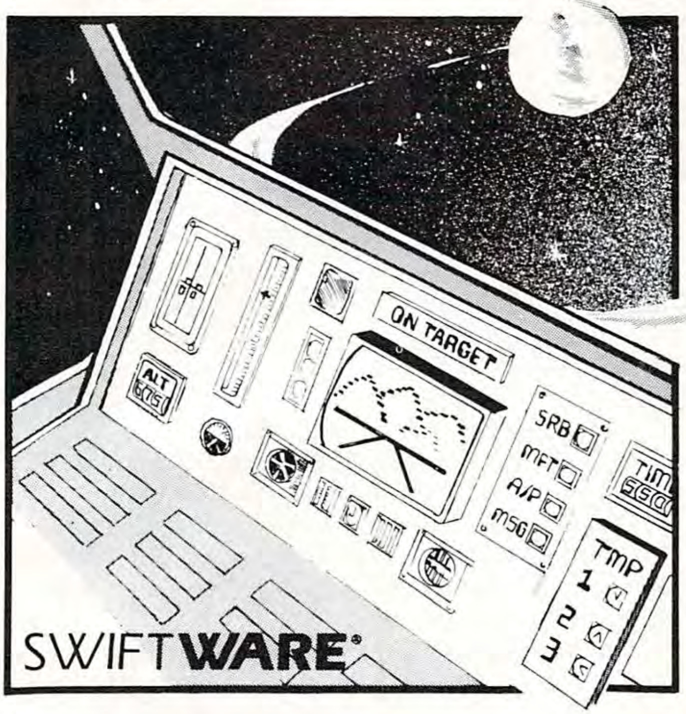
BAIT Statements

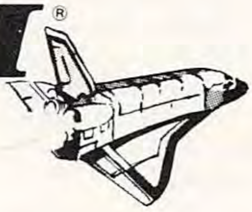
Remember: only the first character of each statement/command name is significant, so what I am really presenting here is a list of which letters of the alphabet we are going to use. The table below lists the first letter, the mnemoic I am using, the syntax of the statement, and (in parentheses) the Atari BASIC equivalent, if indeed that BAIT statement is not the same.

A	**Accept <variable>**	**(INPUT)**
B	**Begin**	**(RUN)**
C	**Call <line-number>**	**(GOSUB)**
D	**Display**	**(LIST)**
E	**End**	
F	**Fetch <address>, <variable>**	**(pseudo-PEEK)**
G	**Goto <line-number>**	
I	**If <expression>, <statement>**	
L	**Let <variable> = <expression>**	
N	**New**	
P	**Print <string-literal>**	
	Print <variable>	
	Print	
R	**Return**	
S	**Store <address>, <expression>**	**(POKE)**

A few of the statements need explanation, which is given below. Also, note that line-numbers and addresses, as used in the above syntax, may

Fly the SPACE SHUTTLE from your ATARI®





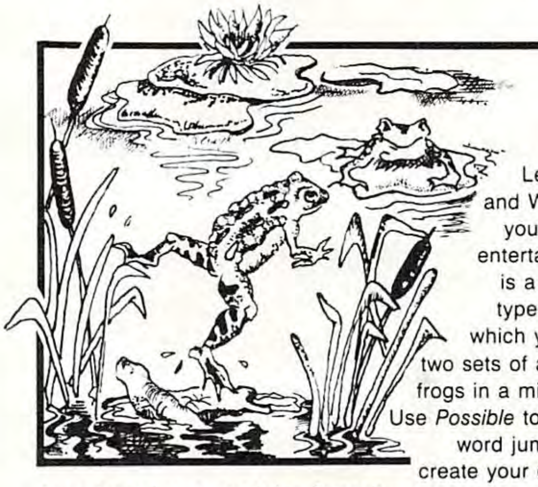
SPACE SHUTTLE

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.



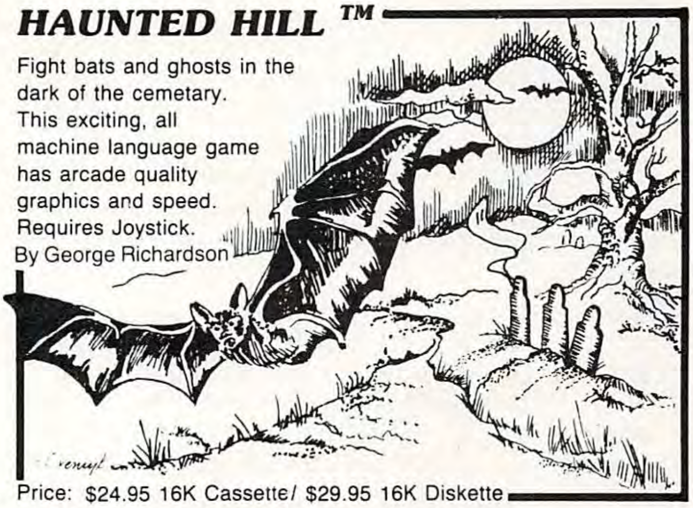
FUN 'N' GAMES™

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™



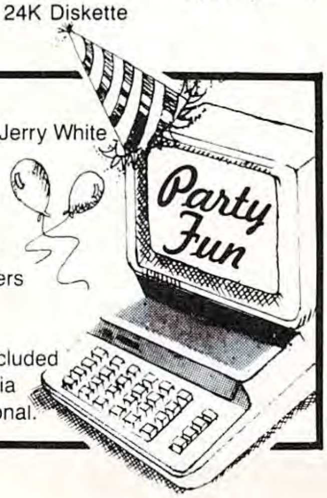
Fight bats and ghosts in the dark of the cemetary. This exciting, all machine language game has arcade quality graphics and speed. Requires Joystick.

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

always be general expressions.

"Accept" allows only a single variable per use, unlike "INPUT" which allows several variables separated by commas.

"Fetch" and "Store" are complementary statements, both with the form of Atari BASIC's "POKE." The only difference is that "Fetch" obviously needs a variable (instead of an expression) to place the fetched (PEEKed) byte into.

"If" does *not* use a "THEN" keyword. Instead, any BAIT statement may follow the comma.

"Let" is a *required* keyword in BAIT. Actually, you may have already presumed this, since otherwise there is no way to distinguish a statement letter from a variable letter in such an assignment statement.

"Print" allows only one item to be printed per statement. Not shown in the above syntax, but allowed by BAIT, are the trailing semicolons or trailing commas, which have the same meaning as under Atari BASIC.

A discussion of what constitutes a valid expression, as well as several other more esoteric points, will have to wait for following month(s).

General Concepts

Since the code for BAIT will be presented in pieces over the course of several months, we must start with a coherent scheme. Also, since we will *not* reprint this month's code next month (for example), the listings must merge properly and neatly.

To this end, I have designated several line number ranges for specific purposes, as listed below.

1000-1999	Initialization of variables used as constants; dimension of strings and arrays; etc.
2000-2999	The "ready" prompt. Get a line of program/command. Parse line for line number.
3000-3999	Program editing. Delete and insert lines.
4000-4999	Control execution of running program. Execute next line, execute command line, etc.
5000-5999	Major subroutine which evaluates arbitrary arithmetic expressions by executing them.
8000-9999	Various miscellaneous subroutines, used by one or more statements.
10000 up	Execution of the actual statements and commands of BAIT. Line numbers of execution routine for each statement are defined in initialization segment, above.

Sidelight: What are the major differences between this scheme and that actually used by the authors of Atari BASIC? (1) There is no provision for generalized I/O routines. (2) Atari BASIC checks the syntax of each line as it is entered and tokenizes it into internal form right then and there. BAIT simply stores exactly what you type in. (3) BAIT is missing many, many of BASIC's capabilities, as noted above.

This Month's Listing

This program is my offering for this month. It consists primarily of the program editor, including the initialization need thereby.

One note about some temporary code: In the finished BAIT, lines 4000 through 4999 will control which statement/command will be executed next. In the case of a command (direct statement, in Atari parlance), these lines will pass control back to the ready prompt when the particular command executor returns. For program editing, we really only need one command, "Display" (LIST), so we have provided a very simple execute control which assumes that *all* direct statements are a request for "Display."

And now for some commentary on the code. Each section of comment is preceded by the line number (or range of numbers) that it refers to.

1010. I chose a practical number here. The larger MAXLINE is, the slower the line deletion process, and the larger the memory you will need. But feel free to change it.

1020. BUFFER$ is used to hold the program you type in and can be almost any size, but be careful: I have not put any provisions in the current BAIT code for detecting when you run out of space.

1030. This is a departure from Atari BASIC (and an effective, though memory-consuming one). Rather than scanning through the program space (BUFFER$) for a line, we "know" where it is via a table kept in LINES.

2360. Since I can't suppress the question mark which the INPUT on line 2300 produces, it is possible that using the Atari cursor keys will sometimes cause the "?" prompt to appear at the beginning of an input line. This gets rid of it by moving the right hand part of the string to the left. (It really works! Try it. And it's also used in line 2720.)

2520 and 2630. Remember, a completed FOR/NEXT loop exits with the loop variable already changed to the first failing value (thus LL + 1 in this example).

2710. If we don't do this, and if LP is greater than LL (i.e., if there is nothing following the line number), then the reference to LINE$(LP) in line 2720 gives us a string length error.

3020. Necessary, if we stripped off the line number.

3040. Shame on you. You typed in a line number with a decimal point, trying to fool me. Gotcha.

3060. The only error message in this month's code.

3110. If the line doesn't yet exist, we can't delete it.

3120, 3130. The number stored in the "LINES"

MORE SERIOUS SOFTWARE FOR THE ATARI®

FORECASTER $49.00

Forecast future events based on past information. Forecast profits, costs, sales trends, prices, test scores, virtually anything. Edit, save on disk and test various elements to determine the outcome. FORECASTER is a powerful "what if" program – a must for business.

STAT PLUS $49.00

STAT PLUS is the most powerful statistic tool we have seen. Run sample space, mean, variance and standard deviation. Do probabilities using binomial, poisson or normal distribution. Also, do students' t-test, Mann-Whitney U test and Chi Square. Do linear regressions; may use 1-12 independent variables. Interact a sample from one module to another, edit, rank and print out.

MASTER LIST $39.00

MASTER LIST is preformatted disk based mail list system. MASTER LIST is designed for use by businesses but can easily be used for individuals. This program features high speed disk to disk sort. Printing formats include continuous list, labels or envelopes. Files can be printed; all the files from one file number to another; by zip code; by area code; by state or by selected files.

LETTER WRITER $39.00

LETTER WRITER is a preformatted letter writing program. LETTER WRITER can be used for any number of applications involving entering, editing and printing text. LETTER WRITER is designed to be easy to use and does not require extensive training. While LETTER WRITER is not a full word processing system, it performs 90% of the functions used by harder to use and more expensive word processors.

LETTER SYSTEM $79.00

This disk contains both LETTER WRITER and MASTER LIST plus the necessary program that allows for combining the two programs for interaction. Call a file off disk, type a letter and (with the right printer) type on envelope, or just run one letter after another. All of the powerful features of both LETTER WRITER and MASTER LIST plus the ability to use them together.

BILLING/INVENTORY $49.00

Invoices, posts to account card, deducts from inventory, enters price and extends. Provisions for shipping, discounts, no charge items and tax allows maintenance for up to 500 customer files. Prints daily total of billings. Inventory allows for up to 300 files. Prints inventory amounts, price list and need list. 2 disk drives - 40K - sheet feed printer.

CALL OR SEND REQUEST FOR OUR FREE CATALOG.

The Programmers Workshop
5230 Clark Ave., Suite 23
Lakewood, CA 90712
Phone Orders (213) 804-1475

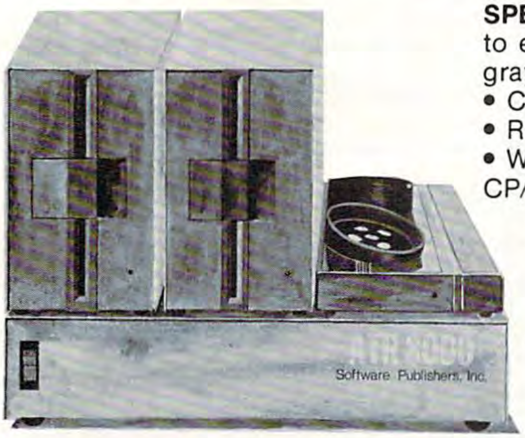
TO ORDER: VISA/MasterCard, check or money order accepted. If charge, please include expiration date of card. Add $1.50 for shipping and handling. California residents add sales tax. Phone or mail.

® ATARI is a registered trademark of Warner Communications

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 (with CP/M)	$250.00	5¼" Drive Cable	$35.00
		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.

table is the length of the line as stored in "BUFFER$" added to 1000 times its starting position in "BUFFER$". We could have used two arrays (one for starting position and one for length) to make it neater, but it would have used a lot more memory.

3140. This line might not work, thanks to a bug in Atari BASIC. Perhaps next month we will have a fix to work around the bug. In the meantime, small programs in BAIT will always work. (Same as the my-system-went-away-when-I-deleted-a-line problem in Atari BASIC.)

3160-3180. This is tricky. After you remove a line via 3140, the starting position of all lines above it in the buffer must be adjusted downward by the size of the line deleted. Can you follow line 3170? Remember, "START" and "LENGTH" refer to the former start and length of the deleted line.

3210. In case we typed in just a line number.

3220-3240. Notice that each new line overlays the "*" which we tack onto the end of the buffer. We then have to put the "*" back on the end. This insures that line 3140 will always work properly, even when we delete the last line in the buffer.

3250. See the comments about lines 3120 and 3130.

3310. If it wasn't a direct line, assume it was added to the program and go after another line.

10100-10150. We check all possible line numbers to see if they need to be listed. Note the similarity between this code and the code needed to delete a line (lines 3110 through 3130): in both cases we need the starting position and length of the line.

10190. Note how each statement will simply RETURN to the execute control code.

Still with me? Go try it. Type it in *very* carefully, backing yourself up every 20 lines or so. If it doesn't work, go back and examine what you typed in, because I guarantee that it worked just seconds before I made this listing for **COMPUTE!**.

Next month, we will try our hand at adding Execute Expression (the most complicated part of what is left) and Print (so we can verify that expressions are executing).

```
1000 REM ..INITIALIZATION..
1001 REM ..................
1010 MAXLINE=99
1020 DIM BUFFER$(5000),LINE$(128)
1030 DIM LINES(MAXLINE)
1040 FOR LP=0 TO MAXLINE:LINES(LP)=0:NEXT LP
1050 BUFFER$="*"
1500 REM LINE NUMBERS OF EXECUTION ROUTINES
1510 PROMPT=2100:INNEXT=2300
1550 DODISPLAY=10100
2000 REM ..INTERACTION..
2001 REM ...............
2100 PRINT "READY"
2300 INPUT LINE$
2350 IF LEN(LINE$)=0 THEN GOTO INNEXT
2360 IF LINE$(1,1)="?" THEN LINE$=LINE$(1):
     GOTO 2350
2370 LL=LEN(LINE$)
2500 REM CHECK FOR LINE NUMBER
2510 FOR LP=1 TO LL
2520 IF LINE$(LP,LP)<="9" AND LINE$(LP,LP)>
     ="0" THEN NEXT LP
2550 REM LP HAS POSITION OF FIRST NON-NUMER
     IC CHARACTER
2560 CURLINE=0
2570 IF LP>1 THEN CURLINE=VAL(LINE$(1,LP-1))
2600 REM NOW SKIP LEADING SPACES, IF ANY
2610 IF LP>LL THEN 2700
2620 FOR LP=LP TO LL
2630 IF LINE$(LP,LP)=" " THEN NEXT LP
2699 REM
2700 REM REMOVE LINE NUMBER AND LEADING SPA
     CES
2710 IF LP>LL THEN LINE$="":GOTO 3000
2720 LINE$=LINE$(LP)
3000 REM ..EDITING..
3001 REM ...........
3010 REM IF HERE, LINE NUMBER IS IN CURLINE
3020 LL=LEN(LINE$):REM AND LL IS LENGTH THE
     REOF
3030 IF CURLINE=0 AND LL=0 THEN GOTO PROMPT
3040 IF CURLINE<>INT(CURLINE) THEN 3060
3050 IF CURLINE<=MAXLINE THEN 3100
3060 PRINT "***BAD LINE NUMBER***"
3070 GOTO PROMPT
3100 REM FIRST, DELETE CURLINE IF IT ALREAD
     Y EXISTS
3110 LENGTH=LINES(CURLINE):IF LENGTH=0 THEN
      3200
3120 START=INT(LENGTH/1000)
3130 LENGTH=LENGTH-1000*START
3140 BUFFER$(START)=BUFFER$(START+LENGTH)
3150 LINES(CURLINE)=0
3160 FOR LP=1 TO MAXLINE:TEMP=LINES(LP)
3170 IF TEMP>=START*1000 THEN LINES(LP)=TEM
     P-LENGTH*1000
3180 NEXT LP
3200 REM NOW ADD LINE TO END OF BUFFER
3210 IF LL=0 THEN GOTO INNEXT
3220 START=LEN(BUFFER$)
3230 BUFFER$(START)=LINE$
3240 BUFFER$(LEN(BUFFER$)+1)="*"
3250 LINES(CURLINE)=START*1000+LL
3300 REM NOW LINE IS IN BUFFER...WHAT DO WE
      DO
3310 IF CURLINE THEN GOTO INNEXT
3320 REM **** TEMPORARY: JUST FALL THROUGH ~
     TO 4000 ****
4000 REM ..EXECUTE CONTROL..
4001 REM ...................
4010 GOSUB DODISPLAY
4020 BUFFER$(INT(LINES(0)/1000))="*"
4030 LINES(0)=0
4040 GOTO PROMPT
4050 REM **** 4010 THRU 4050 ARE TEMPORARY ~
     ****
5000 REM ..EXECUTE EXPRESSION..
5001 REM ......................
8000 REM ..MISCELLANEOUS SUBROUTINES..
8001 REM .............................
10000 REM ..EXECUTE THE VARIOUS STATEMENTS..
10001 REM ..................................
10100 REM ==EXECUTE DISPLAY==
10110 FOR LP=1 TO MAXLINE
10120 LENGTH=LINES(LP):IF LENGTH=0 THEN 1015
     0
10130 START=INT(LENGTH/1000):LENGTH=LENGTH-1
     000*START
10140 PRINT LP;" ";BUFFER$(START,START+LENGT
     H-1)
10150 NEXT LP
10190 RETURN
```

FIRST and FINEST

In Systems Software for Atari and Apple

MAC/65

First we delivered Atari's Assembler/Editor (the cartridge).

Then we produced our enhanced "EASMD."

Now OSS is introducing the **finest** integrated assembly language development system yet!

In addition to being ideal for writing small, "quick and dirty" subroutines and programs, **MAC/65** shows its full power and speed when used with even the most complex of large assembly language source files.

Naturally, **MAC/65** is upward compatible with both EASMD and the Atari cartridge. And, of course, the object code output is also compatible with OS/A+, Atari DOS, and/or Apple DOS, as appropriate.

MAC/65 .... **$80.00***

OS/A+

Optimized Systems Software — the group that produced **both** the **first** Apple DOS **and** the **first** Atari DOS — now brings you OS/A+, which combines the **finest** features of these and other successful personal computer operating systems.

OS/A+ is the **first** and **finest** operating system available for **both** Apple II and Atari computers and features a keyboard-driven, easy-to-use command processor. In addition to several simple resident commands, OS/A+ allows logical and readable requests for even the most sophisticated utility commands. In fact, the user can even add system commands as desired.

But the real power and flexibility of OS/A+ is its ability to easily interface to devices and **disk drives** of virtually any kind and size. File compatibility (with Apple DOS or Atari DOS, as appropriate), device independence, batch processing, easy of use — OS/A+ truly brings the **finest** in operating systems to **your** computer.

AND NOW OS/A+ (for standard Atari or Apple drives) is included as a part of **every** standard OSS language package. Versions of OS/A+ for some higher capacity drives available at extra cost.

Unless otherwise noted, all OSS products require 48K and at least one disk drive. We recommend 64K for the Apple version of OS/A+.

SpeedRead+

The **first** and still **finest** speed reading tutor designed for **you** to use on **your** computer is available **only** from OSS.

SpeedRead+ uses time-proven techniques to train **you** to instantly recognize words and phrases, and yet it goes far beyond what mere mechanical devices are capable of.

SpeedRead+ exercises your peripheral vision, improves your eye movement and timing, and generally works **with you** at your pace... now and in the future.

NOTE: The Atari version of SpeedRead+ needs only 16K of RAM.

SpeedRead+ **$59.95**

tiny C

As a product of Tiny C Associates, **tiny-c** was the **first** structured language interpreter for microcomputers. Now OSS brings this innovative interpretive language to **your** home computer. While not having the speed and power a true **C** compiler, **tiny-c** is an excellent choice for the programming student who is ready to begin learning the valuable techniques of structured languages.

tiny-c provides an easy-to-use, easy-to-modify environment that encourages experimentation while promoting proper programming style. The **tiny-c** package includes not only a comprehensive and instructional user manual but also **complete source.**

tiny-c **$99.95***

C/65

NOW AVAILABLE!

The **first** native mode **C** compiler ever produced for Atari **and** Apple computers.

C/65 supports a very usable subset of the extremely powerful and popular **C** language. Just as **C** is used by the most sophisticated programmers from the professional and academic communities, so shall **C/65** prove to be a powerful and much-needed tool for 6502 software developers.

C/65 supports integer and character types (and arrays), pointers, fully recursive functions, and much more.

NOTE: C/65 requires MAC/65 or an equivalent assembler. Two disk drives recommended but not required.

C/65 .... **$80.00***

BASIC A+

"From the authors of Atari BASIC..."

It's a fact! OSS gave you that **first** and **most popular** language for Atari Home Computers. But why be content with the **first** when you can have the **finest**?

BASIC A+ is the **only** logical upgrade available to the Atari BASIC programmer. While retaining **all** the features which make Atari BASIC so easy to use, we've **also** given BASIC A+ features that place it at the forefront of modern interpretive languages. BASIC A+ will let **you** explore the worlds of structured programming, superior input/output, helpful programming aids, and even a very comprehensive PRINT USING command. **And**, exclusively for the Atari computer, an almost unbelievable array of PLAYER/MISSILE GRAPHICS commands and functions.

BASIC A+ **$80.00***

***REMEMBER:** Standard OS/A+ is **included** at no extra charge with BASIC A+, MAC/65, C/65, and tiny-c.

ATARI, APPLE II, and TINY C are trademarks of Atari, Inc., Apple Computer, Inc., and Tiny C Associates, respectively. ***SpeedRead+***, MAC/65, C/65, BASIC A+ and OS/A+ are trademarks of Optimized Systems Software, Inc.

Optimized Systems Software, Inc., 10379 Lansdale Ave., Cupertino, CA 95014, (408) 446-3099

MACHINE LANGUAGE

Jim Butterfield, Associate Editor

Part I
Numeric Input

It's relatively easy to input strings in machine language. You must receive the characters and put them away neatly. But numbers are a different problem: the ASCII characters must be changed to binary and gathered into a single number.

It's usually best to gather the digits into a buffer rather than to try to process them as they arrive; in this way, you can cope with special characters such as delete and backspace. When the user signals that the input is complete (usually by pressing RETURN), your program can go to the buffer and work out the whole number.

Single Digits

One-digit numbers are fairly easy. If we understand that values coming in from keyboard or file are in ASCII, we're well on the way to doing the job.

ASCII represents the character zero as hexadecimal 30, decimal 48. To print the character zero in BASIC, you'd need to say PRINT CHR$(48). This may seem confusing to beginners (PRINT CHR$(0) doesn't print anything), but it works out well when you get used to it. So hex 30 represents zero; and, if we wish to do arithmetic on it, we must change it to binary zero. The easiest way to do this is with an AND command: AND #$0F will knock out the unwanted high bits.

This works on all the decimal digits: zero, hex 30, up to nine, hex 39. We should check each input character to insure that it is indeed a legitimate digit – otherwise, we may be converting a nonsense character, such as a comma.

Before we output, we must convert our binary value back to ASCII. If its value may be printed as a single digit (0 to 9), the job is once again easy. We simply use the ORA function to insert the missing bits back in: ORA #$30 changes binary to an ASCII digit.

Let's write a simple program to accept a single numeric digit. We'll use $FFE4 for GET, and $FFD2 for PRINT – this will work on all PET/CBM machines, VIC, and Commodore 64. Our coding goes:

```
TOP  JSR   $FFE4   (get a character)
     CMP   #$30    (less than zero ASCII?)
     BCC   TOP     (it's less, go back)
     CMP   #$3A    (greater than nine ASCII?)
     BCS   TOP     (it's greater, go back)
     JSR   $FFD2   (echo to screen)
     RTS           (return to BASIC)
```

We have not converted our number to binary – just checked it to insure that it's in the right range. If our program were to continue, it might perform AND #$0F to convert to binary, and then store the value in A.

As a matter of amusement, let's convert the above program to BASIC POKEs and run it. Our BASIC equivalent goes:

```
100 DATA 32,228,255, 201,48, 144,249
110 DATA 201, 58, 176,245, 32,210,255,
120 DATA 96
200 FOR J=848 TO 862:READ X: POKE J,X:NEXT J
300 FOR J=1 TO 10:SYS 848:NEXT J
```

The first three lines give the machine language program in decimal. The individual instructions have been separated by spaces to make them more visible. Line 200 POKEs the program into the cassette area. Finally, line 300 invokes the machine language program ten times; you will be required to type ten numeric digits. If you try to type other keys, alphabetic or punctuation, the computer will ignore you.

Hexadecimal Input

Hex input is fairly easy. Since each digit is weighted at 16 times the following one, we need to multiply by 16, and that's easy to do, since 16 is a power of two. For example: to convert hex 1234, we must start with the one, multiply by 16, add the two, multiply by 16, add the three, multiply by 16, and finally add the four. If we did this on a calculator, we'd get 4660 as the result. Even though we're working in binary, we must do the same kind of calculation. Let's input four digits and convert them to a binary value. First, a subroutine to get a hex digit in ASCII and convert to binary 0-15:

```
HEXIN JSR  $FFE4   (get a digit)
      TAX          (save a copy)
      CMP  #$3A    (less than 9?)
      BCS  BIG     (no, skip next)
      SBC  #$2F    (convert 0-9)
  BIG CMP  #$41    (A or more?)
      BCC  SMALL   (no, skip next)
```