leagues who have actually researched the matter find that video arcades do *not* cause perversion, or even acne.

After devoting several chapters to his observations of the purported ills foisted upon us by computers, Dr. Brod does give some careful thought to ways to make us capable of handling this technology, even though many of us never knew we had any problems.

Technostress is a book filled with quotable material, and it will probably be heralded as an important book by technophobes everywhere. It will probably receive a lot of press, and its author will probably be in great demand as a speaker, as he carries his message to the world. After all, as a society, we always seem to favor the bad news over the good, and seem to devote our energies to looking for only the real or imagined wrongs in our world.

If Dr. Brod wanted to perform a service to mankind, he might have devoted his energies to solving *this* problem, rather than extrapolating the quirks of his patients to the rest of society. ©

Glenn M. Kleiman

The Computer Speaks, But Will It Listen?

Computer-generated speech, already used in some software, will be incorporated into many educational programs in the next few years. Spoken instructions and responses will be used in programs designed for prereading children and for students who have reading difficulties. Speech will be an integral component of programs which help students learn reading, spelling, and foreign languages, and will make many other types of educational programs more interesting and enjoyable.

Computerized speech can open new worlds for handicapped people. Special programs enable blind users to direct a speech synthesizer to read aloud the words on the computer screen. This makes computerized information bases, word processing, programming languages, and many other computer tools available to the blind. Computerized speech can also help provide communication aids for people with speech impairments.

Computerized speech *recognition* devices are also becoming less expensive and more readily available. These enable computers to recognize words people say, and can make programs easier to use and more appealing. More importantly, speech recognition devices make computers accessible to many people who have physical handicaps which prohibit them from using keyboards.

Two Types Of Computer Speech

There are two general types of computer-generated speech: *stored vocabulary* and *unlimited vocabulary*.

Dr. Glenn M. Kleiman is an educational psychologist and software developer. He is the author of Brave New Schools: How Computers Can Change Education *(Reston/Prentice-Hall) and the designer of* Square Pairs, *an educational game program (Scholastic, Inc.).*

Stored vocabulary speech is created by a person saying the words. Special devices and programs measure characteristics of the sound waveform (for example, intensity, pitch) as the person pronounces each word. Numbers representing the waveform at each fraction of a second are stored in the computer. That is, the speech waveform (an example of what is called *analogue* information) is converted to a sequence of numbers (*digitized* information). The numbers are then used to recreate the sound of the word whenever it is needed.

Stored vocabulary speech can sound very human when individual words are produced. However, it usually sounds choppy and somewhat artificial when the words are combined into sentences. With this technique, the computer is limited to the words previously stored in its memory.

Each digitized word requires a large amount of memory—many numbers must be stored for the computer to recreate the spoken words clearly—so the vocabulary of a personal computer with digitized speech is limited. However, the possibilities for digitized speech will expand as larger-capacity computer memories become less expensive, and as more efficient techniques are developed for representing speech waveforms within the computer's memory.

Unlimited Vocabulary

With unlimited vocabulary speech, programs for generating the individual speech sound (phonemes) are stored in the computer, along with the rules for combining them into words, phrases, and sentences. This technique of speech synthesis enables the computer to produce any word from its component sounds. Synthesized speech does not sound as natural as digitized speech, but it has been greatly improved in recent years.

Phoneme synthesis techniques have been combined with *text-to-speech conversion* programs. These programs contain a set of rules which tell the computer how to change any sequence of letters into speech. Creating a program of this sort for English is difficult, since many letters and letter patterns are pronounced in various ways, depending on the context of their use. For example, the word *read* is pronounced differently depending upon whether it refers to the past or future (for example, *John read the book* versus *John will read the book*). The same aspects of English which cause difficulties for people in learning to read also cause difficulties in programming computers to convert written English to spoken English.

While text-to-speech programs do not produce human-sounding speech, most people understand it easily after a short time—much the way we can understand someone who has a foreign accent and mispronounces some words. Text-to-speech is valuable for people with impaired vision. However, it is not suitable for educational applications in which clear speech is essential.

A Talking Apple

The Echo II speech synthesizer, for Apple II computers, makes use of both stored and unlimited vocabulary techniques. The Echo II is a board that plugs into a slot in the Apple. A speaker or headphone then plugs into the board. The board has volume and pitch controls, but these can also be controlled from software. The basis of the Echo II is a speech synthesis chip made by Texas Instruments. This chip, an advanced version of the one used in the original Speak and Spell toy, is used in most of the available speech synthesizers.

The Echo II comes with a text-to-speech program. It also allows you to enter speech more directly by using symbols to represent each sound (for example, there are different symbols for the long *e* of *Pete* and the short *e* of *bet*). In addition, a disk containing 700 digitized words is available. These provide a good demonstration of the superior quality of digitized speech.

With the Echo II, it is easy to add speech to your own program. You can change the volume, pitch, and rate of speech, all under the control of your program. Produced by Street Electronics, the Echo II sells for about $150. Street Electronics also produces speech synthesizers for the IBM PC and for other personal computers. Other speech synthesizers are available, including Type-'N-Talk from Votrax, Mockingboard from Sweet Micro Systems, and *S.A.M.* from Don't Ask Computer Software.

Computers That Listen

A great deal of research has been devoted to getting computers to recognize people's speech. This research has shown that speech is very complex and that we do not fully understand how people are able to recognize spoken words so easily. It is much more difficult to make computers recognize spoken words than it is to make them pronounce words. However, advances have been made and some usable, although limited, devices are now available.

Current systems for personal computers require the user to program the computer to distinguish among a number of spoken words. The technique is related to stored vocabulary speech. The individual selects a vocabulary to be used. He says each word, then the computer digitizes the sound patterns and stores a set of numbers representing the waveform of the word.

Once trained, the computer recognizes a spoken word by digitizing it and comparing the resulting pattern of numbers to the patterns stored in its memory. Since the pronunciation changes slightly each time an individual says a word, exact matches are not expected, but the computer is programmed to find the closest match. Since people differ widely in their speech patterns, these systems are reliable only in recognizing the words spoken by the person who spoke the original training set.

The digitized representation of each word uses up a lot of computer memory, and the matching process becomes progressively slower and less reliable as more words are added. Therefore, speech recognition systems work well only with limited vocabularies.

It Takes Dictation

One speech recognition device is the Voice Entry Terminal (VET-2), produced by Scott Instruments for Apple II computers. The VET-2 can be programmed for sets of up to 40 words. The Apple II can hold only one set in memory at a time, but others can be loaded from disk as needed.

One important characteristic of the VET-2 is that it functions as a keyboard emulator. It plugs into the computer in parallel with the keyboard, so both can be used together. Each spoken word is associated with a string of printed characters.

When the spoken word is recognized, the VET-2 sends the same signals to the computer that the keyboard sends when the associated keys are pressed. Therefore, you can have the VET-2 recognize a spoken name for each key and then "type" by saying the names of letters, numbers, and special characters. You can then use standard software with voice input replacing the keyboard.

What About Language?

Current technology for personal computers enables us to have computers speak and recognize individual words. But what about sentences and paragraphs? For speech production, we can have the computer string words together, but replicating the intonation and stress patterns of human voices is another, much more difficult, matter.

For speech recognition, anything more complex than the simplest sentence creates inordinate difficulties. Try listening to fluent speakers of a language you do not understand. Can you even tell where one word ends and the next begins? Recognizing the words in spoken sentences generally depends upon being able to understand meanings, something we have not yet learned to program personal computers to do.

Getting computers to produce and understand language is the focus of much of the effort of researchers in artificial intelligence. They have had only limited success, with very powerful computers. For the present, we will have to be content with personal computers which are at the single-word state of language development.

Street Electronics (Echo II)
1140 Mark Ave.
Carpinteria, CA 93013

Sweet Micro Systems (Mockingboard)
Cranston, RI 02910

Votrax (Type-'N-Talk)
500 Stephenson Highway
Troy, MI 48084

Don't Ask Computer Software (Software Automatic Mouth)
2265 Westwood Blvd
Los Angeles, CA 90064

Scott Instruments (Voice-Entry-Terminal)
1111 Willow Springs Drive
Denton, TX 76201

©

INSIGHT: Atari

Bill Wilkinson

This month we'll conclude our exploration of the source code of the program to load a binary file starting with the GET routine presented last month.

Lines 600–619. GET is a special routine for two reasons. First, it assumes a buffer length of zero, thus forcing a single byte transfer into the A register (Atari I/O spec). Second, if the GET fails, it pops a level off the subroutine stack and goes directly to the end-of-file code at line 4000 (BASIC's line 400). This is a crude but effective simulation of the TRAP 400 code in the BASIC version.

For GET to be a general-purpose subroutine, it would have to simply return the status and let the caller do the error trapping.

The Main Section

This routine begins the real work. All object code is reasonably close to its BASIC parallel.

Lines 1200–1204, 1300–1304. Remember the calling requirements for the I/O routines? Channel in X, address in A and Y. Looks easy once you have built the subroutines.

Lines 1400–1407. Same as above. The only extra here is the need to specify a mode for OPEN. Here, we use mode 4 (just as in BASIC) to indicate we will only read the file.

Lines 2400–2405. Since we just stored the A register in HIGH, we test HIGH first by comparing A with 255. If HIGH is equal to 255, then A contains 255 and we can compare it to LOW. A tiny bit sneaky. Did you ever realize that BASIC has to implement THEN this way? By branching around the following code?

Lines 2600–2701. We used LOW and HIGH to get the START address, but we have already moved their contents to START. Now, they won't be used again until we are through with QUIT, so why not share memory between QUIT, LOW, and HIGH? Again, a little bit sneaky, but not inordinately so.

We could have saved more memory (and code) by doing GETs into the low and high bytes of START directly, but I wanted to keep the code as close as reasonable to the original BASIC.

Lines 3100–3106. See the comments above about START and ADDR.

The FOR Loop

Lines 3300–3302. Remember, if a zero page location points to a desired memory location, use an offset of zero in the Y register to store, load, add, etc., to or from that location.

Lines 3403–3408. Since we are STEPping by one, we need check only for equality.

Lines 3411–3417. If the FOR loop had used a STEP, we would have had to add it on here. Since the step is implied to be one, we can use this simple two-byte increment.

Line 4103. If this routine is called from DOS or from BASIC, the RTS is all that is needed, thanks to the POP in the GET routine.

As I said, one could write this routine in better ways. The most obvious thought that comes to mind is to replace the FOR loop with a block get of the requisite bytes. Since that would produce significantly faster runtime (for large files, at least), we will make these changes next month.

To do so, though, we will also change the BASIC program to enable it to make a call to do block I/O. So, even if you are not into machine language, watch next month for a method of doing fast memory reads and writes to and from disk.

Load A Binary Object File—Program Completed

```
Ø66Ø
Ø99Ø BEGINWORK
Ø991 ;
1ØØØ ; BASIC: REM binary object file loader
1ØØ1 ; ---- just a comment ----
11ØØ ; BASIC: DIM NAME$(3Ø)
11Ø1 ;   (the NAME buffer is defined above)
12ØØ ; BASIC: PRINT "WHAT FILE TO LOAD ";
```

ATARI

ATARI is a trademark of ATARI, INC.

ATARI 800XL . Call
Atari 1050 Disk Drive . . . . Call
Atari 1010 Recorder 77.00
Atari 1027 Printer . Call
Atari 850 Interface . . . . Call

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

Replacement Keyboard for Atari 400 . 54.95

3rd Party Printer Interfaces
Interfast I 139.00
Apeface w/cable included 69.95

PRINTERS
Axiom AT-100 . . . . 229.00 — Gemini 10X . . . . Call
Axiom AT-550 . . . . 329.00 — Alphacom . . . . Call
Epson . . . . Call — Okidata 92A . . . . Call
Call for assorted Printer Packages

DISK DRIVES
Rana 1000 . . . .
Trak . . . . Call
Concorde . . . . For
Indus GT . . . . Prices

TOUCH TABLETS
Koala Touch Tablet-D . . . 69.95
Koala Touch Tablet-Cart 74.95

RS232 Modem Adaptor
(Through serial port) 39.95

MONITORS
USI . . . . Call
AMDEK . . . . Call

STIMUTECH SUBLIMINAL SOFTWARE FOR ATARI AND CBM 64
Expando-Vision Interface w/one FREE Cart . . . 99.00
Weight Control
Study Habits
Stress Control
Drinking Control
Smoking Control
Career Success
Sexual Confidence
Addit'l Rom Carts 29.95 ea.

ATARI SOFTWARE

ATARI
Conversational Lang-T . . 44.95
My First Alphabet-D 26.95
Home Filing Mgr.-D 37.95
Star Raiders-Cart 32.95
Assembly Editor-Cart . . . 46.95
Macroassembler-D 67.95
Invitation to Program I-T . 19.95
Basketball-Cart 26.95
Graph-it-T 15.95
Dig Dug-Cart 32.95
Atariwriter-Cart 74.95
Donkey Kong-Cart 37.95
Ms. Pac-Man-Cart 39.95
Eastern Front-Cart 32.95
Donkey Kong Jr.-Cart . . . 39.95
Pengo-Cart 35.95
Logo-Cart 79.95
Robitron-Cart 35.95
Pole Position-Cart 39.95
Microsoft Basic II-Cart . . 67.95
Paint-D 33.95
Caverns of Mars-Cart . . . 32.95
Joust-Cart 39.95
Jungle Hunt-Cart 35.95
Mario Bros.-Cart 35.95
Syncalc-D 74.95
Atari Music-D 29.95

SIERRA ON-LINE
Homeword-D 49.95
Dark Crystal-D 27.95
Frogger-D/T 23.95
Quest For Tires-D 23.95
Ultima I-D 23.95
Ultima II-D 41.95
Wizard/Princess-D 22.95

INFOCOM
Deadline-D 34.95
Witness-D 34.95
Zork I, II, III-D 27.95
Starcross-D 27.95
Planetfall-D 34.95
Suspended-D 34.95
Enchanter-D 34.95
Infidel-D 34.95
Sorcerer-D 34.95

LJK
Letter Perfect-D 74.95
Data Perfect-D 74.95
Spell Perfect-D 64.95

D - Disk T - Cassette
Cart - Cartridge

BRODERBUND
Arcade Machine-D 41.95
Bank St. Writer-D 49.95
Lode Runner-D 23.95
Drol-D 23.95
Spare Change-D 23.95
Choplifter-D 23.95
Mask of the Sun-D 27.95
Operation Whirlwind-D . . 27.95

EPYX
Gateway To Apshai-Cart 27.95
Pitstop-Cart 27.95
Fun With Music-Cart 27.95
Jumpman Jr.-Cart 27.95
Dragonriders of Pern-D/T 27.95
Temple of Apshai-D/T . . 27.95
Hellfire Warrior-D/T 20.95
Fun With Art-Cart 27.95
Summer Games-D 27.95
Puzzlemania-D 23.95

SSI
Carrier Force-D 41.95
Combat Leader-D/T 27.95
Cosmic Balance-D 27.95
Cosmic Balance II-D 27.95
Fortress-D 23.95
Rails West-D 27.95
Broadsides-D 27.95
War In Russia-D 54.95
Questron-D 34.95

SPINNAKER
Up For Grabs-Cart 27.95
Cosmic Life-Cart 23.95
Facemaker-Cart 23.95
Alphabet Zoo-Cart 23.95
Fraction Fever-Cart 23.95
Kids on Keys-Cart 23.95
Delta Drawing-Cart 27.95
Story Machine-Cart 27.95
Amazing Things-D 27.95
Hey Diddle Diddle-D 20.95
Snooper Troops 1 or 2-D . 30.95
Trains-D 27.95
Aerobics-D 30.95
Adventure Creature-D . . . 27.95

PARKER BROS.
Astrochase-Cart 34.95
Frogger-Cart 34.95
Q-Bert-Cart 34.95
Popeye-Cart 34.95

MISCELLANEOUS FOR ATARI
Castle-Wolfenstein-D . . . 20.95
Home Accountant-D 52.95
Master Type-D/Cart 27.95
Spelunker-D 27.95
Flight Simulator II-D 37.95
Zombies-D/T 23.95
Mr. Robot-D 23.95
River Raid-Cart 31.95
Diskey-D 34.95
Ultima III-D 41.95
Miner 2049'er-Cart 34.95
Scraper Caper-Cart 34.95
Basic XL-Cart 74.95
Monkey Wrench II-Cart . . 49.95
Omnimon 82.95
Starbowl Football-D/T . . . 22.95
Hockey-Cart 27.95
Flip Flop-D/T 20.95
Boulder Dash-D/T 20.95
Encounter-D/T 23.95
Quasimoto-D/T 23.95
N.Y.C.-D/T 23.95
Compute Books . . . . Call
Pitfall-Cart 31.95
Codewriter-D 69.95
The Heist-D 27.95
Mac 65XL-Cart 74.95
Jupiter Mission-D 34.95
Nibbler-D 20.95
Air Support-D/T 23.95
Pogo Joe-D/T 20.95
Analog Compendium -Book 9.95
Movie Maker-D 41.95
Antica 4-D 34.95

Hundreds of items available for the Atari and CBM 64 . . . please call.

COMMODORE 64

CBM 64 . . . . Call
1541 Disk Drive 239.00
1526 80 Column Printer 279.00
1530 Datasette 66.00
1702 Color Monitor 249.00
1650 AD/AA Modem 89.00
RS 232 Interface 44.00

COMMODORE 64 SOFTWARE

Program Ref. Guide 19.95
Assembler-D 17.95
Easy Finance I, II, II, IV-D 19.95
Easy Calc-D 64.95
Easy Mail-D 17.95
Easy Script-D 39.95
Easy Spell-D 19.95
Logo-D 49.95
The Manager-D 37.95
General Ledger-D . . 37.95
Accts. Rec.-D 37.95
Accts. Pay.-D 37.95
Magic Desk-D 52.95
Zork I, II or III-D 29.95
Suspended-D 29.95
Starcross-D 29.95
Deadline-D 29.95

COMMODORE 64 SOFTWARE

ACCESS
Neutral Zone-D/T 23.95
Spritemaster-D/T 27.95
Beachhead-D/T 23.95
Music Composer-D 27.95

HES
Multiplan-D 69.95
64 Forth-Cart 41.95
HES Writer-Cart 32.95
HES Mon-Cart 27.95
Turtle Graphics-Cart . . . 41.95
Mutant Camels 20.95
HES Modem 59.95
Super Zaxxon . . . . Call

SYNAPSE
Blue Max-D/T 23.95
Zaxxon-D/T 27.95
Morgul-D/T 23.95
Shamus II-D/T 23.95
Sentinal-D/T 23.95
Pharoah's Curse-D/T . . 23.95
Slamball-D/T 23.95
Zepplin-D/T 23.95

CARDCO
Cardprint/A 59.95
Cardprint/B 37.95
Cardco + G 64.95
Cardboard/5 . . . . 54.95
Cardkey 29.95
Cardette/I 29.95
Cardwriter/I 29.95
Cassette Recorder 47.95
Printer Utility-D/T 19.95
Write Now-Cart 34.95
Mail Now-D 27.95

HANDIC
64 Forth-Cart 29.95
64 Graf-Cart 23.95
Stat 64-Cart 23.95
Calc Result Easy-Cart . . 34.95
Calc Result Advanced-D 74.95
The Diary-Cart 23.95
The Tool-Cart 29.95

CBS SOFTWARE
Call for items and prices.

SPINNAKER
Trains-D 27.95
Kidwriter-D 23.95
Alf In The Caves-Cart . . 27.95
Jukebox-Cart 23.95

CYMBAL
Accounts Payable-D . . . 52.95
Accounts Receivable-D 52.95
Inventory Control-D 52.95
Invoice Writer-D 52.95

MISCELLANEOUS FOR CBM 64
Flight Simulator II-D . . . 37.95
Night Mission/ Pinball-D 20.95
Praticalc PS-D 59.95
M-File-D 64.95
Word Pro 3+ /Spell-D . . 74.95
Home Accountant-D . . . 52.95
Step By Step-D/T 44.95
Barron's Sat.-D 59.95
Money Manager-D/T . . . 19.95
Bristles-D/T 20.95
Telestar 64-Cart 37.95
Star League Baseball-D/T 20.95
Castle Wolfenstein-D . . 20.95
Sorcerer-D 34.95
Computer Baseball-D . . 27.95
Ringside Seat-D 27.95
Professional Golf-D 27.95
Mastertype-Cart 27.95
Solo Flight-D 23.95
Vic Switch 124.95
Paper Clip-D 64.95
Delphi's Oracle-D 74.95
Super Buscard II . . . . Call
Instaspeed-D/T 99.95
Codewriter-D 69.95
Cyberia Farm Mgr.-D . . . 37.95
First Class Mail-D 34.95
Aztec-D 27.95
Popeye-Cart 34.95
Ultima III-D 41.95
Ultima II-D 41.95
Miner 2049er-Cart 27.95

ACCESSORIES
WICO Joystick . . . . Call
Flip 'n' File Trunks 20.95
Flip 'n' File Cart 20.95
Joysensor 24.95
Elephant Disks (Box of 10) 20.00
WICO Trakball 37.95
KRAFT Joystick 15.95

To Order Call Toll Free
800-558-0003

For Technical Info, Order Inquiries, or for Wisc. Orders -
414-351-2007

New Lower Prices

ComputAbility

NO SURCHARGE FOR MASTERCARD OR VISA

ORDERING INFORMATION. For fast delivery send cashier's check, money order or direct bank transfers. Personal and company checks allow 2 weeks to clear. Charges for COD are $3.00. School Purchase Orders welcome. In CONTINENTAL USA, include $3.00 shipping per software order. Include 3% shipping on all Hardware orders, minimum $3.00. Mastercard & Visa please include card # and expiration date. WI residents please add 5% sales tax. HI, AK, FPO, APO, Canadian orders — add 5% shipping, minimum $5.00. All other foreign orders, please add 15% shipping, minimum $10.00. All goods are new and include factory warranty. Due to our low prices, all sales are final. All defective returns must have a return authorization number. Please call 414-351-2007 to obtain an RA# or your return will NOT be accepted for replacement or repair. Prices and availability are subject to change without notice.

COMPUTABILITY
P.O. Box 17882
Milwaukee, WI 53217

ORDER LINES OPEN
Mon-Fri 11 AM - 7 PM CST
Sat 12 PM - 5 PM CST

```
Ø66Ø A2ØØ      12Ø1      LDX #Ø*16    ; channel Ø is screen
Ø662 A9Ø6      12Ø2      LDA #MESSAGE/256
Ø664 AØØ3      12Ø3      LDY #MESSAGE&255 ; address of message to A,Y
Ø666 2Ø24Ø6    12Ø4      JSR PRINT    ; do the actual work
               13ØØ ; BASIC: INPUT NAME$
Ø669 A2ØØ      13Ø1      LDX #Ø*16    ; channel Ø is screen
Ø66B A9Ø5      13Ø2      LDA #NAME/256
Ø66D AØ8Ø      13Ø3      LDY #NAME&255 ; address of name to A,Y
Ø66F 2Ø29Ø6    13Ø4      JSR INPUT    ; do the actual work
               14ØØ ; BASIC: OPEN #1,4,Ø,NAME$
Ø672 A21Ø      14Ø1      LDX #1*16    ; channel 1
Ø674 A9Ø4      14Ø2      LDA #4
Ø676 9D4AØ3    14Ø3      STA ICAUX1,X ; mode for open
Ø679 A9Ø5      14Ø4      LDA #NAME/256
Ø67B AØ8Ø      14Ø5      LDY #NAME&255 ; address of name
Ø67D 2Ø1AØ6    14Ø6      JSR OPEN     ; again, do the actual work
Ø68Ø 3ØDE      14Ø7      BMI BEGINWORK ; if bad file or name, ask again
               2ØØØ ; BASIC: REM get and check header
               2ØØ1 ;  (just a comment)
               2ØØ2 ;
Ø682           2ØØ3 LINE2ØØ
               2ØØ4 ;
               21ØØ ; BASIC: TRAP 4ØØ
               21Ø1 ;  implemented in code below
               22ØØ ; BASIC: GET #1,LOW : GET #1,HIGH
Ø682 A21Ø      22Ø1      LDX #1*16    ; channel 1
Ø684 2Ø46Ø6    22Ø2      JSR GET      ; get a byte to A reg
Ø687 8D18Ø6    22Ø3      STA LOW
Ø68A A21Ø      22Ø4      LDX #1*16    ; channel 1
Ø68C 2Ø46Ø6    22Ø5      JSR GET      ; get a byte to A
Ø68F 8D19Ø6    22Ø6      STA HIGH
               23ØØ ; BASIC: TRAP 4ØØØØ
               23Ø1 ;  really, just turns trap off
               23Ø2 ;  again, implemented in code below
               24ØØ ; BASIC: IF LOW=255 AND HIGH=255
               24Ø1 ;          GET #1,LOW : GET #1,HIGH
Ø692 C9FF      24Ø2      CMP #255
Ø694 DØ15      24Ø3      BNE EOL24Ø   ; HIGH is not 255
Ø696 CD18Ø6    24Ø4      CMP LOW
Ø699 DØ1Ø      24Ø5      BNE EOL24Ø   ; LOW is not 255
Ø69B A21Ø      24Ø6      LDX #1*16    ; LOW=255 and HIGH=255, so...
Ø69D 2Ø46Ø6    24Ø7      JSR GET      ; get a byte to A reg
Ø6AØ 8D18Ø6    24Ø8      STA LOW
Ø6A3 A21Ø      24Ø9      LDX #1*16    ; channel 1
Ø6A5 2Ø46Ø6    2411      JSR GET      ; get a byte to A
Ø6A8 8D19Ø6    2412      STA HIGH
Ø6AB           2413 EOL24Ø
               2414 ; (the BNE's get here to skip the THEN statements)
               2499      .PAGE ".         BASIC lines 25Ø-41Ø"
               25ØØ ; BASIC: START=LOW+256*HIGH
Ø6AB AD18Ø6    25Ø1      LDA LOW
Ø6AE 85CE      25Ø2      STA START    ; move LOW byte
Ø6BØ AD19Ø6    25Ø3      LDA HIGH
Ø6B3 85CF      25Ø4      STA START+1 ; same as 256*HIGH
               26ØØ ; BASIC: GET #1,LOW : GET #1,HIGH
Ø6B5 A21Ø      26Ø1      LDX #1*16    ; channel 1
Ø6B7 2Ø46Ø6    26Ø2      JSR GET      ; get a byte to A reg
Ø6BA 8D18Ø6    26Ø3      STA LOW      ; LOW byte of QUIT
Ø6BD A21Ø      26Ø4      LDX #1*16    ; channel 1
Ø6BF 2Ø46Ø6    26Ø5      JSR GET      ; get a byte to A
Ø6C2 8D19Ø6    26Ø6      STA HIGH     ; HIGH byte of QUIT
               27ØØ ; BASIC: QUIT = LOW+256*HIGH
               27Ø1 ;  already done by 26Ø1 thru 26Ø6, see text
               3ØØØ ; BASIC: REM read in a segment
               3ØØ1 ;  (just a comment)
```

```
                3100 ; BASIC: FOR ADDR=START TO QUIT
                3101 ;  a small sneaky here:
                3102 ;     START and ADDR are same
                3103 ;     location in ass'y language version
                3104 ;
06C5            3105 FORADDRLOOP
                3106 ;
                3200 ; BASIC: GET #1,BYTE
06C5 A210       3201    LDX #1*16   ; channel 1
06C7 204606     3202    JSR GET     ; get a byte to A-reg
                3203    ;             notice that we do not actually store it
                                      yet
                3300 ; BASIC: POKE ADDR,BYTE
06CA A000       3301    LDY #0      ; needed for indirect addressing
06CC 91CE       3302    STA (ADDR),Y ; an effective poke
                3400 ; BASIC: NEXT ADDR
                3401 ;    strangely, this simple BASIC statement
                3402 ;    causes a lot of work in ass'y language
06CE A5CE       3403    LDA ADDR
06D0 CD1806     3404    CMP QUIT    ; at end of loop yet?
06D3 D007       3405    BNE DONEXT  ; no
06D5 A5CF       3406    LDA ADDR+1
06D7 CD1906     3407    CMP QUIT+1  ; try high bytes also
06DA F008       3408    BEQ ENDOFFOR ; aha! not yet
06DC            3411 DONEXT
06DC E6CE       3412    INC ADDR    ; change low byte of addr
06DE D0E5       3413    BNE FORADDRLOOP
06E0 E6CF       3414    INC ADDR+1  ; and high byte if needed
06E2 D0E1       3415    BNE FORADDRLOOP
                3416 ;
06E4            3417 ENDOFFOR
                3500 ; BASIC: GOTO 200 : REM try another segment
06E4 4C8206     3501    JMP LINE200
                4000 ; BASIC: REM trapped to here, assume end of file
                4001 ;    (just a comment)
06E7            4002 LINE400
                4100 ; BASIC: CLOSE #1
06E7 A210       4101    LDX #1*16   ; channel 1
06E9 201F06     4102    JSR CLOSE   ; do the work
06EC 60         4103    RTS         ; no parallel in BASIC,
                4104 ; we have to RTS to operating system
```

©

64 EXPLORER

Larry Isaacs

A complete drawing package should allow the user to print characters on the bitmapped display. This month and next I will discuss this topic, and give more examples on the use of the drawing routines presented last month.

There are two methods for printing characters on a bitmapped display. We can POKE the dot patterns of the characters to the bitmapped RAM, or we can draw the characters onto the display.

Let's take a look at the first method, which is faster because no line-drawing routines are required.

POKEing To The Bitmap

The first step in POKEing characters to a bitmapped display is to choose the *cell size*, or dimensions, of our character set. The choice of the cell size can greatly affect the complexity of the routines which print the characters. If a convenient size is chosen, the routines will be simplified; if you are up for a challenge, you can write the routines to accept a variable cell size.

We will use a cell size of 8 dots high by 8 dots wide, for two reasons. First, a width of 8 dots is the number of dots which can be held in a byte. Second, there already exists a set of 8 x 8 characters in the 64's character ROM.

Actually, the 64's normal character display mode is very similar to what we want to accomplish in a bitmapped mode. The process involves character cells and some method of transferring the character dot patterns to the display. However, in the normal character display mode, the format is determined by the character codes found in a character array called screen memory. In screen memory, you can change only whole 8 x 8 characters, so your effective resolution winds up being 40 columns by 25 lines.

Blending Characters And The Bitmap

When using a bitmap display, you can control each dot. This implies that you can place a character at any X,Y position on the screen. This can certainly be done, though it is more difficult than placing a character code in screen memory. What complicates the task somewhat is that the 64's VIC-II chip organizes a bitmap display as groups of 8 x 8 dot cells.

It's possible for the 8 x 8 dot character pattern to span as many as four of the bitmap cells, two horizontally and two vertically. This doesn't create much of a problem vertically, but horizontally the bytes in the character dot pattern may have to be moved or shifted to span two bytes. In addition, when the bytes are added to the bitmap, the routine must not disturb the dots outside the shifted 8 dots of the character pattern.

Next, we must decide how to transfer the dot patterns so they will be visible against the bitmapped background.

Using Conditional Logic

One way of transferring the dot pattern is to add (logical OR) the dots in the pattern to the dots already in the display. Dots which are on in the character dot pattern are also turned on in the display. Dots which are on in the display remain on. This avoids erasing the background as a character is printed to the bitmapped display, but can result in illegible characters if there are too many dots already turned on in the background.

Another way to transfer the dot pattern is to flip (Exclusive-OR) the dots in the pattern into the bitmapped RAM. Dots in the bitmapped RAM which correspond to on dots in the dot pattern are flipped to the opposite state. The advantage of this technique is that it will make characters visible regardless of whether the background is on or off. However, characters can still be illegible if the background is not predominantly either on or off.

Or the transfer could be accomplished by writing the pattern directly into the bitmapped RAM. This type of transfer replaces the background with the character cell. We will use this technique.

A BASIC Example

Let's first demonstrate how the required routines might be implemented in BASIC. Unfortunately,

like the drawing routines presented in earlier columns, the character routines are too slow to be really useful. To enhance their value as an example, we'll try to illustrate modular programming style as well.

One of the main aspects of modular programming is breaking main or primary tasks into smaller, more manageable tasks. Once the tasks have been broken down sufficiently, each may be implemented in a single routine. The more independent each of these separate routines is, the better. This allows you to concentrate on the details involved with the routine as it is written, without being distracted by the details involved with other routines. To show how printing to the bitmapped display might be broken into modules, let's take a look at the logical subdivisions of this task.

Although this program isn't really complex enough to justify a modular approach, I prefer to keep the functions or tasks in separate routines, so long as the routines don't become embarrassingly simple. This helps while debugging, since the symptoms of the bug often eliminate a majority of the routines from consideration. It also helps keep you from accidentally tangling functions together.

When functions get tangled or intertwined, making one change may require making other changes, leading to a snowball effect. And finally, it is good practice to keep functions divided into separate routines when you write a complex program. How well the tasks are divided up can greatly affect how much effort it takes to write and debug the program.

Breaking Down The Task

Putting a character in a bitmapped display will involve transferring bytes into bitmapped RAM, so we will need a routine which does the transferring. We need another routine to calculate the character's position. We also need to know what to write; this will require two routines. We need a routine which will find and read the appropriate bytes in the character ROM. However, the dot patterns are organized based on screen codes, which are different from the Commodore 64 ASCII codes you normally print. This means we need a routine to convert the ASCII code to the corresponding screen code.

Finally, we need a routine to do the horizontal shifting necessary when the character byte needs to span two bytes in the bitmapped RAM. This gives us five routines to be implemented:

1. Convert the character to screen code
2. Read the character's dot pattern
3. Calculate its position in the bitmap and amount of shift
4. Shift a dot-pattern byte
5. Put the dot-pattern byte in the bitmap

By dividing the tasks into well-defined and independent sections, it will be a little easier to implement them than if you tried to throw it all together in one routine. For example, converting the ASCII character code to a screen code can be done without concerning ourselves with where the ASCII code came from, or for what the screen code will be used. The shift section does not need to account for where the shift amount came from or what will be done with the shifted bytes.

Combining The Modules

Once we build the character print routine from these five sections, it is simple to build a string print routine using the character print routine. The result might be a BASIC program like the one that accompanies this article. The program uses the machine language routines discussed in this column in previous issues. Before running this program, you must run the BASIC loader presented in the May 1984 issue. The subroutine at line 100 converts ASCII code CH to the equivalent screen code SC. The subroutine at line 200 uses screen code CH to read the associated dot pattern into the array DP(). This subroutine also uses CP which points to the base of the character dot patterns in ROM.

The subroutine at line 300 uses the coordinates in X and Y to calculate an offset OF into the bitmap, and the shift amount SH. The subroutine at line 400 uses the shift amount SH to right-shift the byte in BY partially into B2. This means shifting dots out of the right end of the byte and into the left end of the other byte. This shift routine also makes the mask bytes, M1 and M2.

Finally, the subroutine at line 500 writes the bytes into the bitmap base of the offset OF, calculated earlier. This routine also uses the mask bytes to keep the necessary old bits from the bitmap bytes, before adding the new dot pattern bits. The subroutine at line 600 prints the character at the current coordinates specified by X and Y, and the subroutine at 700 prints a string at X,Y.

Logical Math

I have used logical operators (OR and AND) rather than division and the INT functions. For example, in line 320, the term (X AND −8) gives the same result as INT(X/8)*8. In the subroutine at line 200, the POKEs are required to turn off interrupts and make the character ROM accessible to the BASIC program.

The main routine uses the string-printing routine at line 700 to label the vertical axis for the plot of a sine wave. As you will see, the character printing is pretty slow. This part of the program would be much more useful written in machine language. Next month I will discuss the drawing method of putting characters in the bitmapped display, and present machine language routines for both.

Characters On A Bitmapped Display

```
10 REM PRINT CHARACTERS TO BIT-MAP:rem 63
20 JV=49152:REM JUMP TABLE          :rem 6
30 CP=53248:REM LOC. OF CHAR. PATTERNS
                                 :rem 181
40 POKE 785,PEEK(JV+28):REM SETUP USR()
                                   :rem 8
50 POKE 786,PEEK(JV+29)           :rem 17
60 GOTO 1000                      :rem 96
100 REM CONVERT CHAR. TO SCREEN CODE
                                  :rem 96
110 IF CH>31 AND CH<64 THEN SC=CH:RETURN
                                 :rem 249
120 IF CH>63 AND CH<96 THEN SC=CH-64:RETU
    RN                           :rem 155
130 IF CH>95 AND CH<128 THEN SC=CH-32:RET
    URN                          :rem 200
140 IF CH>127 AND CH<192 THEN SC=CH-64:RE
    TURN                         :rem 251
150 SC=CH-128:RETURN             :rem 214
200 REM GET CHARACTER DOT PATTERN:rem 232
210 POKE 56334,PEEK(56344) AND 254
                                 :rem 221
220 POKE 1,PEEK(1) AND 251        :rem 50
230 FOR IX=0 TO 7                :rem 100
240 DP(IX)=PEEK(CP+SC*8+IX):NEXT :rem 202
250 POKE 1,PEEK(1) OR 4          :rem 159
260 POKE 56334,PEEK(56334) OR 1   :rem 69
270 RETURN                       :rem 121
300 REM CALC OFFSET AND SHIFT COUNT
                                  :rem 43
310 TY=199-Y:SH=X AND 7           :rem 27
320 OF=(TYAND-8)*40+(XAND-8)+(TYAND7)
                                 :rem 106
330 RETURN                       :rem 118
400 REM SHIFT BYTE TO CORRECT POSITION
                                  :rem 84
410 B2=0:M1=0:M2=255:IF SH=0 THEN RETURN
                                  :rem 13
420 FOR K=1 TO SH:B2=B2/2         :rem 52
430 IF BY AND 1 THEN B2=B2 OR 128 :rem 85
440 BY=BY/2:M1=(M1/2)OR128:M2=M2/2:NEXT
                                  :rem 28
450 RETURN                       :rem 121
500 REM PUT BYTE AT X,Y           :rem 24
510 GOSUB 300:REM CALCULATE OF & SH
                                 :rem 171
520 GOSUB 400:REM SHIFT OVER     :rem 131
530 AD=57344+OF:REM GET ADDRESS FOR BY
                                 :rem 167
540 POKE AD,USR(OF) AND M1 OR BY :rem 230
550 IF SH=0 THEN RETURN           :rem 64
560 POKE AD+8,USR(OF+8) AND M2 OR B2
                                 :rem 136
570 RETURN                       :rem 124
600 REM PUT CHARACTER AT X,Y     :rem 114
610 GOSUB 100:REM CONVERT CH     :rem 114
620 GOSUB 200:REM READ DOT PATTERN
                                 :rem 233
630 Y=Y+8:REM PUT CHAR. FROM TOP DOWN
                                 :rem 173
640 FOR IX=0 TO 7:Y=Y-1:BY=DP(IX):rem 136
650 GOSUB 500:REM PUT BYTE       :rem 251
660 NEXT:RETURN                  :rem 245
700 REM PUT STRING S$ AT X,Y      :rem 52
710 FOR SP=1 TO LEN(S$)          :rem 218
720 CH=ASC(MID$(S$,SP,1))        :rem 123
730 GOSUB 600:REM PUT THE CHARACTER
                                  :rem 53
740 X=X+8:NEXT                   :rem 100
750 RETURN                       :rem 124
1000 REM MAIN ROUTINE            :rem 240
1010 SYS JV:SYS JV+6,0:SYS JV+9,0,1
                                 :rem 237
1020 FOR I=0 TO 10               :rem 100
1030 LB=-1+I*.2:S$=STR$(LB)      :rem 213
1050 X=5:Y=46+10*I:GOSUB 700:NEXT:rem 147
1060 SYS JV+12,32,50:SYS JV+18,32,150
                                 :rem 214
1070 SYS JV+12,32,100:SYS JV+18,319,100
                                  :rem 54
1080 FOR I=0 TO 10               :rem 106
1090 X=30:Y=50+10*I              :rem 246
1100 SYS JV+12,X,Y:SYS JV+18,X+4,Y
                                 :rem 205
1110 NEXT                          :rem 2
1120 SYS JV+12,32,100:PI=3.1416  :rem 124
1130 SX=256/(2*PI):SY=50          :rem 71
1140 FOR I=0 TO 2*PI STEP 2*PI/100
                                 :rem 236
1150 SYS JV+18,32+I*SX,100+SIN(I)*SY
                                  :rem 22
1160 NEXT                          :rem 7
9000 GET Z$:IF Z$="" THEN 9000   :rem 231
9010 SYS JV+3                    :rem 199
```

©

Jim Butterfield, Associate Editor

Decimal Mode
Part 1

The 6502 has an option which affects only the add (ADC) and subtract (SBC) instructions: decimal mode.

Decimal mode is invoked with the Set Decimal (SED) command, and canceled with Clear Decimal (CLD). It may be affected by stack activities that pull the status register—PLP for Pull Processor status, and RTI—but this is unusual. In most computer environments you can assume that decimal mode is not in force when your program is invoked; but if you're not sure, it won't hurt to give a CLD.

Decimal mode is intended to help with certain types of numbers: Binary Coded Decimal (BCD) numbers. You might want to use this type of number system when the values are used mostly for input and output with little calculation involved.

Binary numbers—the computer's usual numeric values—are good for advanced calculations. Multiplication and division are easy to do in binary, and more advanced calculations can readily be developed. The only problem with binary numbers is this: They must be converted to decimal at the time of input or output.

Decimal numbers, or more accurately BCD numbers, are easy to input and output since they are held in the same decimal notation as was entered or will be seen by the user. With decimal mode, we may add or subtract these numbers without converting them to binary. But if we want to do more advanced mathematics, we'll certainly go to binary.

Accounting programs often use decimal mode. Similarly, many games keep scores in decimal format, since the only activities are adding points as they are scored and displaying the results.

What Is BCD?

The easiest way to describe a number held in Binary Coded Decimal is this: When you display it in hexadecimal format, you see the correct decimal value. Let's explain this with a few examples.

A value of 9 is held within a byte as binary 00001001. This is true whether you are using binary or BCD numbering. If we print the contents of this byte in hexadecimal, it is displayed as 09. Now, this not only represents the value nine, it looks like nine.

If we are in binary mode and add one to the above value, we'll get 00001010. The value is ten but the number displays in hex as 0A. This doesn't look like ten to those of us who are not trained to read hex. Worse: If we add six, we'll get a value of 16, which prints as hex value 10. This doesn't look like 16—if we didn't know it was a hexadecimal number, we might think it was ten.

Let's go back to our original value of nine, but switch to decimal mode. If we add one, using the ADC instruction, we'll end up with binary 00010000. We know that the value must represent ten, and when we print the hexadecimal it shows up as 10—which looks like ten. We must ignore the usual binary rules, which would tell us that binary 00010000 is equivalent to decimal 16. In BCD, this binary number has a value of 10. If we add a six in decimal mode, we'll get 00010110 which has a value of 16 and prints out as hexadecimal 16.

We've decided to use the bits in a different way. The four high bits—the high nybble, as it's sometimes called—represent a tens digit; the four low bits, or low nybble, represent units. Each nybble may have a value from 0 to 9, but the six

highest combinations corresponding to hex A, B, C, D, E, and F will never be used.

This makes BCD less efficient than binary for storing numbers. The highest BCD number that we can store within a single byte is 99, as compared to 255 for binary. We can use several bytes together to hold larger numbers, but BCD always holds less: A two-byte BCD number can go from 0000 to 9999, compared to a two-byte unsigned binary number which can range from 0 to 65535.

But it's convenient. When we wish to output such a number, we extract each digit, convert it to ASCII with an ORA #$30, and print it. (We get the left digit by using four LSR instructions, and the right digit with AND #$0F.) An equivalent binary number would need a divide-by-ten routine before it could be output.

Similarly, input is a snap. As each ASCII digit arrives, it has its high bits stripped (with AND #$0F) and gets packed together with another digit to generate the two-to-a-byte BCD value.

An Example

Here's a sample program to show the power of BCD numbers and ease of programming with them. We'll have the computer (PET, VIC, or 64) output a table of multiples of the number 142857. This is a favorite peculiar number of mine; you'll see why when we print the table.

```
                        ; set value to zero
033C  A2  00            LDX  #$00
033E  8E  90  03             STX  LOW
0341  8E  91  03             STX  MED
0344  8E  92  03             STX  HIGH
                        ; do the addition
0347  18                LOOP CLC
0348  78                     SEI
0349  F8                     SED
034A  AD  90  03             LDA  LOW
034D  69  57                 ADC  #$57
034F  8D  90  03             STA  LOW
0352  AD  91  03             LDA  MED
0355  69  28                 ADC  #$28
0357  8D  91  03             STA  MED
035A  AD  92  03             LDA  HIGH
035D  69  14                 ADC  #$14
035F  8D  92  03             STA  HIGH
0362  D8                     CLD
0363  58                     CLI
                        ; print the number
0364  A0  02                 LDY  #$02
0366  B9  90  03        LP   LDA  LOW,Y
0369  4A                     LSR  A
036A  4A                     LSR  A
036B  4A                     LSR  A
036C  4A                     LSR  A
036D  09  30                 ORA  #$30
036F  20  D2  FF             JSR  $FFD2
0372  B9  90  03             LDA  LOW,Y
0375  29  0F                 AND  #$0F
0377  09  30                 ORA  #$30
0379  20  D2  FF             JSR  $FFD2
037C  88                     DEY
037D  10  E7                 BPL  LP
                        ; print RETURN and loop
037F  A9  0D                 LDA  #$0D
0381  20  D2  FF             JSR  $FFD2
0384  E8                     INX
0385  E0  07                 CPX  #$07
0387  D0  BE                 BNE  LOOP
0389  60                     RTS
```

Note that we hold the value we are calculating in three bytes, called LOW, MED, and HIGH; we add starting at the low byte and working up. The Carry flag works the same way as is usual for addition. While we're in decimal mode, we lock out the interrupt so that the interrupt routines won't do their arithmetic in the wrong mode. The addition sequences could have been written as a loop; for the sake of clarity, it was done using "straight line" coding.

For printing, we start from the high byte, of course. The output routine for BCD is simple compared to what we would need to do with binary values.

If you'd rather enter the program from BASIC, here's the same program in DATA statements. It will work on all Commodore machines.

```
100 DATA 162,0,142,144,3,142,145,3
110 DATA 142,146,3,24,120,248,173,144,3
120 DATA 105,87,141,144,3,173,145,3
130 DATA 105,40,141,145,3,173,146,3
140 DATA 105,20,141,146,3,216,88,160,2
150 DATA 185,144,3,74,74,74,74,9,48
160 DATA 32,210,255,185,144,3,41,15,9,48
170 DATA 32,210,255,136,16,231,169,13
180 DATA 32,210,255,232,224,7,208,190,96
200 FORJ=828 TO 905
210 READX:T=T+X
220 POKEJ,X
230 NEXT J
240 IF T<>8325 THEN STOP
250 SYS 828
```

You might like to examine the output of the program to see what's so special about the first seven multiples of the number 142857.

Next month, we'll discuss special features and wrinkles of decimal mode. ©

Atari Artist

Andrew S. Katz

With this program, you can place shapes of any size, orientation, or color anywhere on the screen. Use the joystick to create the shape, and change its color with the press of a single key.

In spite of its simplicity, "Atari Artist" can be used to draw complex designs as well as realistic scenes. Draw a circle inside a triangle inside a circle, and so on. Piece together a house in the midst of a forest. Then store your art on disk or tape.

Atari Artist comes in two versions. Version 1, a four-color version, has a blue status window and runs in 16K. Version 2, a 16-color GTIA version, has a gray status window and needs 24K to run.

To use this program, you'll need a joystick plugged into port 1. Be sure to have a cassette recorder or disk drive attached if you wish to LOAD and SAVE copies of your designs. When you first RUN the program, the title screen will appear. It will give you information about the keys' uses. At this point, you may wish to select a version. Version 1 is set up by default. If you wish to use Version 2, press and release the joystick button. The number 2 should replace the 1 after the word version. Press the button again to return to Version 1.

Once you've selected the version, move the joystick. If you have selected Version 2 and the message ERROR 147 ON LINE 1000 is on the screen, your Atari does not have enough memory for Version 2. Type RUN again, and this time use Version 1.

Marking The Shape

After several seconds the play screen will appear. All three markers are on top of one another at the top of the screen. Notice the two-line status window at the bottom of the screen.

To move a marker, push the joystick in the direction you want the marker to go. It should respond instantly. The marker you are moving is called the current marker and is indicated by a pinkish tint. The other two markers are white. The markers may move anywhere on the screen, including the hidden area behind the status window. If you try to move it off the screen, the marker will stop at the screen's boundary.

To control the other markers, release the joystick and press the joystick button. Notice that MARKER # lights up in the status window. This is to show you that you are in the process of picking a new current marker. Release the joystick button. MARKER # is no longer lit up, but the number beneath it has changed. It has increased by one, or cycled back from 2 to 0. Also, a different marker now has a pinkish tint. That's the marker that now responds to the joystick. Very soon, you will find the movement of the markers and the switching between them to be quite simple.

Change The Marker Speed

The speed at which the joystick moves the markers across the screen can be changed. Speeds range from 1 to 9. Speed 1 is normal, Speed 2 is twice as fast, and so on up to Speed 9, which is nine times as fast as Speed 1. The higher speeds do not permit you to stop at every point on the screen. These high speeds are used to get across the screen quickly, or to assist in more advanced drawing. To change speed, press the joystick button and move the joystick.

Notice that the highlighted item in the status window changed from MARKER # to SPEED. Move the joystick toward you to decrease speed or away from you to increase speed. You will see the number under SPEED in the status window change as you move the joystick. When you've reached the desired speed, release the joystick button. Now when you use the joystick, it will move the current marker at the speed you set.

To change the color, release the joystick and press the OPTION key. Notice that in the status window COLOR has lit up. This is to show you that you are in the process of choosing the next color in the sequence. When you release the OPTION key, the next color is shown beneath COLOR. To step through the color sequence, repeatedly press and release OPTION. When the color sequence reaches the last color, it starts again from the first color (the one in effect when you first started). Each version has its own color sequence listed in the table. The colors you actually see may vary, depending upon your computer and the tint adjustment on your TV.

Drawing Colors

Version 1:

- ORANGE
- GREEN
- BLUE
- BLACK or erase

Version 2:

- GOLD
- ORANGE
- REDORG (red orange)
- PINK
- PURPLE
- VIOLET
- STBLUE (steel blue)
- BLUE
- BYBLUE (baby blue)
- TURQUO (turquoise)
- GRBLUE (green blue)
- GREEN
- YELGRE (yellow green)
- ORGGRE (orange green)
- LTGREN (light green)
- BLACK or erase

Two Fundamental Shapes

To change the shape, release the joystick and press the SELECT key. Notice that in the status window SHAPE has lit up. This is to show you that you are in the process of selecting the other fundamental shape. The two fundamental shapes are TRIANGL (triangle) and CIRCLE. Now, release the SELECT key. The shape underneath SHAPE has changed from TRIANGL to CIRCLE, or from CIRCLE to TRIANGL.

To draw a shape, just press START. The program will take control and draw the shape. When the drawing has completed, control over the keys and joystick returns to you. The amount of time it takes to draw a shape will depend upon its size. A circle takes more time than a triangle, and Version 2 is slower than Version 1. The shape shown under SHAPE is drawn and given the color shown under COLOR. If the shape is TRIANGL, the three markers are its corners. If the shape is CIRCLE, it is drawn using the markers as points along its circumference. As the shape is drawn, it covers (or erases) anything that was in its position on the screen.

SAVEing The Screen

To store the screen display on cassette or disk, or to reload a previously stored screen, press the OPTION and SELECT keys at the same time. The status window is then replaced with the first level of prompt. If you press RETURN, you'll get the status window back. You must press L for LOAD or S for SAVE. Other keys will be rejected and a buzz will sound. Do not press BREAK or SYSTEM RESET.

When you press S or L, the second level of prompt will be shown. Now, you must type a filename such as C for cassette or D:ANDY.GRT for disk. You cannot type more than 15 letters for a filename. Any additional letters or invalid keystrokes will be ignored. Mistakes can be corrected with the backspace key. After typing a filename, press RETURN. If no filename is shown, you will get the status window back. If the filename is invalid, you will see ERROR DETECTED TRY AGAIN for several seconds before the status window returns.

If the filename was correct and you have the disk or cassette set up, the SAVE or LOAD should proceed as explained in the tape or disk manual. When the SAVE or LOAD is complete or interrupted, the status window returns.

Keep The Versions Separate

During a LOAD, the second prompt will come with a warning to use files saved under the current version. A Version 2 screen loaded into Version 1 will result in some striped colors and height distortion. A Version 1 screen loaded into Version 2 will result in different colors and height distortion. Also, it will attempt to LOAD past the end of the file. During a LOAD you will see the screen fill from top to bottom.

Feel free to interrupt a LOAD by pressing BREAK. This is a way to merge the top of a SAVEd screen with the bottom of the current screen. However, pressing BREAK or SYSTEM RESET may cause the program to crash. If this happens, press SYSTEM RESET and type RUN.

Before drawing the shape, the program calculates the numbers it needs from the positions of the markers. For the purpose of positioning, the screen is treated as an X-Y grid with X,Y pairs for each separate point or pixel on the screen. The X can be thought of as column and the Y as row. The upper left corner of the screen is assigned 0,0 and the lower right corner is assigned 159,79 (79, 159 in V2). Then it uses the numbers to draw the shape one row at a time.

Creating A Triangle

Lines 507–540 contain the triangle predrawing section. Line 510 finds the highest (A), middle (B), and lowest (C) markers by comparing the markers' Y coordinates. Line 1550 has the six possibilities for three markers listed out in advance. Lines 530–536 calculate the slopes of the imaginary lines connecting the markers. Lines 11–30 contain the drawing routine. There are two sections divided by a horizontal line at B. In the first section, horizontal lines are drawn from line CA to line BA. In the second section, horizontal lines are drawn from line CA to line CB. The two special cases where AY = BY or BY = CY are also handled.

The circle predrawing section is lines 600–680. The two crucial factors here are the location of the center of the circle (RX,RY), and the radius of the circle (R). The center of the circle is found by using the bisectors rule from geometry. To apply that rule, connect points C and A and points B and A. Then, make lines which pass through the midpoints of lines CA and BA and are perpendicular to CA and BA. We can use the point-slope method to describe these lines. Finally, find where these lines intersect. That is done by solving simultaneous linear equations.

Plotting A Circle

To find the radius, calculate the distance from the center of the circle to point A. In the program, any of the three markers are used as points A, B, and C. The markers are tried in different orders in line 1550 until a center is found.

Notice line 650. The TRAP is there to test for the case where the slopes of the bisectors are equal. This will occur only when the three markers are in a straight line. You can't draw a circle on a straight line. The actual drawing is performed by lines 2–10. It is done by drawing the upper half and the bottom half simultaneously, starting at the equator and going to the poles. X,Y pairs which are outside the screen range are converted to fit on the screen for partial horizontal lines.

Finally, an FT factor is used to make round circles. If you draw circles without using FT, they come out oval. This is because the height of a screen pixel is not equal to its width.

With careful planning, you can construct interesting designs or detailed scenes that have the quality of a watercolor painting. By combining the two fundamental shapes of nature—the circle and the triangle—you can form many other shapes such as rectangles, stars, diamonds, and crescents. The program teaches children drawing composition and the names of the colors.

Drawing A Rectangle

Let's draw a rectangle.

Step 1: Move the markers together until they are exactly on top of one another. This will be the lower left corner of the rectangle.

Step 2: Increase the speed (9 is OK).

Step 3: Move a marker right by tapping the joystick. Count how many taps you make.

Step 4: Do the same thing with another marker but in the up direction.

Step 5: Press START.

Step 6: Move the third marker right and then up the same number of times you counted in steps 3 and 4.

Step 7: Press START and you'll have a rectangle.

Now that you have the general idea, try drawing some shapes on your own.

Here's some advice about circles. Since the markers form the edge of the circle, lining up the markers in a straight line will form a very large circle. In fact, it may not form a circle at all, because you can't draw a curve on a straight line. The computer will buzz at you if you tell it to draw a straight line circle. Move one marker a little and try again. You will see that very large circle. Sometimes circles are partly off and partly on the screen. If the partly off part is drawn first, you may have to wait a few seconds before you see your circle being drawn. Be patient. Soon you will become familiar with how circles are made, so you will know in advance how one will come out before it's drawn.

The Background Comes First

When you draw a scene, remember to do the background first. It is just like painting: The new shape will cover the old. You may notice that certain colors contrast each other and certain colors blend into each other. This and other visual effects can and should be used to your advantage. Remember also that the same color can look different with different backgrounds.

If you see the colors changing after you have been drawing for a while, your Atari is in *attract mode*. The purpose of attract mode is to protect your TV from permanent burn-in of colors. To get your normal colors back, just press the SPACE bar or a letter key.

There is no specific feature for clearing the screen, but it's easy to start with a clean slate. Just move the markers to three of the corners of the screen and draw a BLACK TRIANGL. Then move a marker to the fourth corner from the corner diagonally opposite and draw again.

You may want to modify the program. One simple modification is to use the 16 shades of the GTIA mode. In this mode, the names of the colors should be reinterpreted as shades of gray. In lines 1525 and 1530, change 623 to 65, 87 to 9, and 712 to 0.

Refer to the "Automatic Proofreader" article before typing this program in.

```
FL 1 GOTO 10000
IC 2 FOR Y=0 TO R:X=SQR(RS-Y*Y):X1=FT*(RX-X):X2=FT*(RX+X):Y1=RY-Y:Y2=RY+Y
PO 3 IF Y1>YMAX THEN Y1=YMAX
AB 4 IF Y2>YMAX THEN Y2=YMAX
BH 5 X1=X1*(X1>0):IF X1>XMAX THEN X1=-1
PP 6 IF X2>XMAX THEN X2=XMAX
OC 7 IF X1<0 OR X2<0 THEN 10
PE 8 IF Y1>=0 THEN PLOT X1,Y1:DRAWTO X2,Y1
PI 9 IF Y2>=0 THEN PLOT X1,Y2:DRAWTO X2,Y2
PN 10 NEXT Y:GOTO 100
HB 11 IF AY=BY THEN PLOT AX,AY:DRAWTO BX,BY:GOTO 20
IA 15 FOR Y=AY TO BY:PLOT CX-(CY-Y)*KCA,Y:DRAWTO BX-(BY-Y)*KBA,Y:NEXT Y
HI 20 IF BY=CY THEN PLOT BX,BY:DRAWTO CX,CY:GOTO 30
IH 25 FOR Y=BY TO CY:PLOT CX-(CY-Y)*KCA,Y:DRAWTO CX-(CY-Y)*KCB,Y:NEXT Y
CN 30 GOTO 100
NE 100 REM MAIN LOOP
NI 110 IF PEEK(53279)=6 THEN 500
HG 120 ST=STICK(0)
OB 130 IF ST=15 THEN IF PEEK(53279)=3 THEN 700
OD 132 IF ST=15 THEN IF PEEK(53279)=1 THEN 900
OJ 135 IF ST=15 THEN IF PEEK(53279)=5 THEN 800
GF 140 IF ST=15 THEN IF STRIG(0)=0 THEN 300
OD 150 XM(MARKER)=XM(MARKER)+SPEED*X(ST):YNEW=YM(MARKER)+SPEED*Y(ST)
HA 160 IF XM(MARKER)>206 THEN XM(MARKER)=206
BH 170 IF XM(MARKER)<48 THEN XM(MARKER)=48
CE 180 IF YNEW<16 THEN YNEW=16
HP 190 IF YNEW>111 THEN YNEW=111
JD 200 POKE 53252+MARKER,XM(MARKER)
CA 204 IF YNEW=YM(MARKER) THEN 210
OA 205 POKE PMM+YM(MARKER),PEEK(PMM+YM(MARKER))-MK(MARKER):POKE PMM+YNEW,PEEK(PMM+YNEW)+MK(MARKER):YM(MARKER)=YNEW
FN 210 GOTO 100
HM 300 S$(3,10)=HEID$(1,8)
BK 320 IF STICK(0)<>15 THEN S$(3,10)=HEAD$(1,8):GOTO 400
GA 330 IF STRIG(0)=0 THEN 320
OE 345 POKE 704+MARKER,14
AO 350 MARKER=MARKER+1:IF MARKER=3 THEN MARKER=0
OG 355 S$(3,10)=HEAD$(1,8):S$(46,46)=CHR$(MARKER+16):POKE 704+MARKER,60
GD 360 GOTO 100
OL 400 S$(11,18)=HEID$(9,16)
BN 405 FOR W=1 TO 50:NEXT W
EI 410 IF STICK(0)=15 THEN S$(11,18)=HEAD$(9,16):GOTO 100
DH 420 IF STICK(0)=14 THEN SPEED=SPEED+1:IF SPEED>9 THEN SPEED=9
CH 430 IF STICK(0)=13 THEN SPEED=SPEED-1:IF SPEED<1 THEN SPEED=1
PG 435 S$(55,55)=CHR$(SPEED+16)
GK 440 GOTO 405
GN 500 COLOR COLR:RESTORE 1550
MN 505 IF SHAPE=1 THEN 600
OP 507 TRAP 699:READ A,B,C
IJ 510 IF YM(A)<=YM(B) AND YM(B)<=YM(C) THEN AY=YM(A):AX=XM(A):BY=YM(B):BX=XM(B):CY=YM(C):CX=XM(C):GOTO 520
GM 511 GOTO 507
NJ 520 AX=(AX-48)*XT:BX=(BX-48)*XT:CX=(CX-48)*XT:AY=(AY-16)*YT:BY=(BY-16)*YT:CY=(CY-16)*YT
EE 530 TRAP 532:KCA=(CX-AX)/(CY-AY)
EF 532 TRAP 534:KBA=(BX-AX)/(BY-AY)
EP 534 TRAP 536:KCB=(CX-BX)/(CY-BY)
MJ 536 TRAP 40000
DE 540 GOTO 11
OJ 600 TRAP 699:READ A,B,C
DC 601 AX=(XM(A)-48)*XT/FT:AY=(YM(A)-16)*YT
DG 610 BX=(XM(B)-48)*XT/FT:BY=(YM(B)-16)*YT
DL 620 CX=(XM(C)-48)*XT/FT:CY=(YM(C)-16)*YT
II 625 IF CY=AY OR BY=AY OR AX=BX THEN 600
DK 630 KCA=(AX-CX)/(CY-AY)
DJ 632 KBA=(AX-BX)/(BY-AY)
PB 640 LCA=(CY+AY)/2-KCA*(CX+AX)/2
OP 642 LBA=(BY+AY)/2-KBA*(BX+AX)/2
PE 650 TRAP 699:RY=(KBA*LCA-KCA*LBA)/(KBA-KCA)
NN 660 TRAP 40000:RX=(RY-LBA)/KBA
AG 670 RS=(RY-AY)*(RY-AY)+(RX-AX)*(RX-AX)
NP 675 R=SQR(RS):IF R>200 THEN 699
AJ 680 GOTO 2
EF 699 TRAP 40000:FOR I=0 TO 30:POKE 53279,0:NEXT I:GOTO 100
CD 700 S$(19,26)=HEID$(17,24)
OA 720 IF PEEK(53279)=3 THEN 720
GG 730 COLR=COLR+1:IF COLR=NCOLRS THEN COLR=0
AJ 740 S$(19,26)=HEAD$(17,24):S$(60,65)=COLR$(COLR*6+1,COLR*6+6)
GG 750 GOTO 100
CA 800 S$(27,34)=HEID$(25,32)
OE 820 IF PEEK(53279)=5 THEN 820
BI 830 SHAPE=1-SHAPE
NE 840 S$(27,34)=HEAD$(25,32):S$(68,74)=SHAPE$(SHAPE*7+1,SHAPE*7+7)
GH 850 GOTO 100
GA 900 SS$=S$:POKE 764,255:CLOSE #2:OPEN #2,4,0,"K:":POKE 702,64:POKE 694,0
HH 910 S$="  PRESS [L] TO LOAD SCREEN FROM FILE{8 SPACES}PRESS [S] TO SAVE SCREEN TO FILE{8 SPACES}"
JF 920 FOR I=1 TO 80:S$(I,I)=CHR$(ASC(S$(I,I))-32):NEXT I
JO 925 GOSUB 2000:IF A=155 THEN 999
FN 926 IF A=ASC("L") THEN W=4:B=7:GOTO 940
JE 927 IF A=ASC("S") THEN W=8:B=11:GOTO 940
OI 930 FOR I=1 TO 25:POKE 53279,0:NEXT I:GOTO 925
JD 940 S$="  FILE NAME ?{67 SPACES}":FILE$=S$(14,28)
BL 941 IF W=4 THEN S$(42,65)="FILE MUST BE FOR VERSION":S$(67,67)=CHR$(ASC("0")+V)
```

```
JK 943 FOR I=1 TO 80:S$(I,I)=CHR$(ASC(
       S$(I,I))-32):NEXT I
FE 950 I=0
CK 955 GOSUB 2000:IF A=155 AND I=0 THE
       N 999
BH 960 IF A=155 THEN 980
KF 965 IF A=126 AND I=0 THEN 955
EF 968 IF A=126 THEN A=ASC(" "):GOSUB
       978:I=I-1:GOTO 955
ON 970 IF I=15 THEN 955
EI 975 I=I+1:GOSUB 978:GOTO 955
CH 978 FILE$(I,I)=CHR$(A):S$(14+I,14+I
       )=CHR$(A-32):RETURN
IJ 980 TRAP 997:POKE 54286,64
PD 981 IF FILE$(1,1)="C" THEN POKE 537
       75,35:POKE 53768,40:POKE 53764,
       0:POKE 53766,0:POKE 53773,255
HI 982 OPEN #1,W,0,FILE$
JN 985 POKE 852,PEEK(88):POKE 853,PEEK
       (89):POKE 856,0:POKE 857,15*V:P
       OKE 850,B
LB 990 B=USR(ADR(CIO$)):GOTO 999
LL 997 TRAP 40000:POKE 54286,192:S$(42
       ,67)="ERROR DETECTED TRY AGAIN
       "
NM 998 FOR I=42 TO 67:S$(I,I)=CHR$(ASC
       (S$(I,I))-32):NEXT I:FOR I=1 TO
        1000:NEXT I
NC 999 TRAP 40000:S$=SS$:CLOSE #2:CLOS
       E #1:GOTO 1520
BN 1000 IF V=2 THEN GRAPHICS 24:PM=PEE
        K(106)-36:NCOLRS=16:XT=0.5:YT=
        2:FT=0.3125:XMAX=79:YMAX=191
BO 1001 IF V=1 THEN GRAPHICS 23:PM=PEE
        K(106)-20:NCOLRS=4:XT=1:YT=1:F
        T=1.25:XMAX=159:YMAX=95
MH 1003 POKE 54279,PM:PMM=PM*256+384:P
        OKE 559,38:POKE 53277,1
JP 1004 POKE 623,1:FOR I=PMM TO PMM+12
        7:POKE I,0:NEXT I
AD 1005 FOR I=0 TO 2:POKE 704+I,14:NEX
        T I
BH 1010 DIM XM(2),YM(2),MK(2),COLR$(96
        ),SHAPE$(14)
LC 1011 FOR I=0 TO 2:XM(I)=125:POKE 53
        252+I,125:YM(I)=16:NEXT I:POKE
         PMM+16,255
PC 1012 MK(0)=3:MK(1)=12:MK(2)=48:REM
        MISSILE MASKS
KP 1013 COLR$="BLACK GOLD  ORANGEREDOR
        GPINK  PURPLEVIOLETSTBLUE BLUE
         BYBLUETURQUOGRBLUEGREEN YELGR
        EORGGRELTGREN"
PG 1014 SHAPE$="TRIANGLCIRCLE ":IF V=1
         THEN COLR$(1,24)="BLACK ORANG
        EGREEN BLUE  "
IC 1015 FOR I=1 TO 96:COLR$(I,I)=CHR$(
        ASC(COLR$(I,I))-32):NEXT I
PL 1016 FOR I=1 TO 14:SHAPE$(I,I)=CHR$
        (ASC(SHAPE$(I,I))-32):NEXT I
JB 1020 SHAPE=0:COLR=1:MARKER=0:SPEED=
        1:POKE 704,60
JM 1021 DIM S$(80):S=ADR(S$):SH=INT(S/
        256):SL=S-SH*256
JA 1022 S$=" ":S$(80)=" ":S$(2)=S$
MA 1023 FOR I=1 TO 80:S$(I,I)=CHR$(ASC
        (S$(I,I))-32):NEXT I
MF 1025 DIM HEAD$(32):HEAD$="MARKER #
         SPEED  COLOR{3 SPACES}SHAPE
        "
BG 1026 DIM HEID$(32):HEID$="MARKER #
         SPEED  COLOR{3 SPACES}SHAPE
        "
DP 1027 FOR I=1 TO 32:HEAD$(I,I)=CHR$(
        ASC(HEAD$(I,I))-32):NEXT I
FA 1028 FOR I=1 TO 32:HEID$(I,I)=CHR$(
        ASC(HEID$(I,I))-32):NEXT I
DA 1029 S$(3,34)=HEAD$:S$(46,46)=CHR$(
        MARKER+16):S$(55,55)=CHR$(SPEE
        D+16):S$(60,65)=COLR$(COLR*6+1
        ,COLR*6+6)
EC 1030 S$(68,74)=SHAPE$(SHAPE*7+1,SHA
        PE*7+7)
OP 1034 DIM X(15),Y(15):FOR I=5 TO 15:
        READ A,B:X(I)=A:Y(I)=B:NEXT I
LJ 1035 DATA 1,1,1,-1,1,0,0,0,-1,1,-1,
        -1,-1,0,0,0,0,1,0,-1,0,0
EE 1100 DIM CIO$(6):FOR I=1 TO 6:READ
        A:CIO$(I,I)=CHR$(A):NEXT I
MJ 1105 DATA 104,162,16,76,86,228
EH 1110 DIM SS$(80),FILE$(15)
KC 1500 DIM DLI$(14):FOR I=1 TO 14:REA
        D A:DLI$(I,I)=CHR$(A):NEXT I
KC 1501 DATA 72,173,111,2,41,3,141,10,
        212,141,27,208,104,64
DB 1502 DL=PEEK(560)+PEEK(561)*256:IF
        V=1 THEN 1514
AD 1503 POKE DL+182,143:POKE DL+183,66
        :POKE DL+186,2:POKE DL+187,PEE
        K(DL+199):POKE DL+188,PEEK(DL+
        200)
JG 1513 POKE DL+189,PEEK(DL+201):POKE
        DL+184,SL:POKE DL+185,SH:GOTO
        1520
LC 1514 POKE DL+93,66:POKE DL+96,2:POK
        E DL+97,PEEK(DL+101):POKE DL+9
        8,PEEK(DL+102)
AJ 1515 POKE DL+99,PEEK(DL+103):POKE D
        L+94,SL:POKE DL+95,SH:GOTO 152
        0
DO 1520 POKE 513,INT(ADR(DLI$)/256):PO
        KE 512,ADR(DLI$)-256*PEEK(513)
MJ 1521 POKE 54286,192
EJ 1525 IF V=2 THEN POKE 623,193:POKE
        87,11
HI 1530 IF V=2 THEN POKE 712,8:POKE 71
        0,8:POKE 709,14
JE 1540 GOTO 100
ED 1550 DATA 0,1,2,2,1,0,0,2,1,1,2,0,1
        ,0,2,2,0,1
DL 2000 A=PEEK(764):IF A=255 OR A=60 O
        R A=39 THEN 2000
BC 2005 GET #2,A:IF A=126 OR A=155 THE
        N RETURN
PB 2010 IF A<32 OR A>=96 THEN 2000
KI 2015 RETURN
AM 10000 GRAPHICS 17:POKE 752,1:V=1
NE 10001 DL=PEEK(560)+256*PEEK(561)
BI 10002 POKE DL+3,71:FOR I=6 TO 11:PO
         KE DL+I,7:NEXT I
FI 10010 POSITION 7,1:? #6;"WELCOME"
FM 10020 POSITION 10,3:? #6;"to"
DG 10030 POSITION 2,5:? #6;"Shapes and
          ColoRs"
IJ 10040 POSITION 0,10:? #6;"OPtION  C
         HANGE COLORS"
GE 10050 POSITION 0,11:? #6;"select  C
         HANGE SHAPES"
FP 10055 POSITION 0,12:? #6;"START
         {3 SPACES}TO DRAW"
HH 10060 POSITION 0,15:? #6;"move stic
         k to begin";
DI 10065 POSITION 0,14:? #6;"button fo
         r version ";V;
FL 10066 IF STICK(0)<>15 THEN 1000
HM 10067 IF STRIG(0)=0 THEN V=3-V
DD 10068 IF STRIG(0)=0 THEN 10068
DF 10069 GOTO 10065
```

©

C. Regena

Programming Techniques In TI BASIC

This month, by answering some of the common questions I have received from readers, I'm going to give you a variety of programming techniques that you can use in your own programs.

How do you clear part of a screen?

Let's say you have onscreen a nice picture with a description underneath. CALL CLEAR will clear the whole screen; but you want to clear the printing, not the picture. Use CALL HCHAR with the row and column parameters under the picture, and use the number of repetitions that will clear the section you want. For example, to clear the lower half of the screen, CALL HCHAR(13,1,32,32*12). We're starting with row 13, column 1, and clearing with the space (character code 32) for 32*12 squares—32 columns times 12 more rows.

To clear with a different color, redefine a character (in a color set you are not using) as a colored square, then use CALL HCHAR to put that character on the screen:

```
300 CALL COLOR(13,16,16)
310 CALL HCHAR(13,1,128,32*12)
```

To clear a vertical section of the screen, use CALL VCHAR:

CALL VCHAR(1,17,32,24*16)

To try out this technique, try this sample program:

```
100 CALL HCHAR(1,1,42,32*24)
110 CALL HCHAR(13,1,32,32*12)
900 GOTO 900
```

Change line 110 to the CALL VCHAR statement above and try the program. Next take out line 110 and put in lines 300 and 310 listed above. Experiment with different numbers of repetitions.

How do you get a border around the screen?

CALL SCREEN(*c*), where *c* is a number from 1 to 16, defines the screen color. When you use this statement in a program, the whole screen instantly changes color. CALL COLOR(*s*,*f*,*b*) defines the character colors. The characters are divided into sets of eight characters each. The *s* in the parentheses is the set number and can be from 1 to 16. The *f* is the foreground color of the character, *b* the background color, and they can be one of the 16 color numbers, from 1 to 16.

Now take a look at the characters in set 1. The space is code 32 in set 1. The screen is filled with spaces wherever there isn't any printing or graphics. If you change the color of set 1 to something other than the screen color (background color 1), you'll get color where all the spaces are.

```
100 CALL CLEAR
110 CALL SCREEN(14)
120 CALL COLOR(1,2,16)
900 GOTO 900
```

Press FCTN 4 (CLEAR) to stop the program. You've got a border on the top and on the bottom, but you would like the sides also. When we PRINT messages we have a 28-column line, but when we do graphics we actually have 32 columns—there are two columns on each side of the regular printing section. They currently have spaces in them. To get the screen color in those columns, add

```
115 PRINT ::::::::::::::::::::::::
```

Or, as you print messages, those extra columns fill with the screen color. (As you PRINT, columns 1, 2, 31, and 32 will contain character 31.) A quicker way to get rid of the spaces in those columns is to fill the columns with a character in the screen color. You may add these lines instead:

```
115 CALL CHAR(152,"")
116 CALL VCHAR(1,1,152,48)
117 CALL VCHAR(1,31,152,48)
```

Now try a few PRINT messages, such as

```
150 PRINT "HELLO"
```

Notice that the letters have little squares of the screen color around them. All the color sets are automatically defined as CALL COLOR(S,2,1), which is black with a transparent background. The color number 1, transparent, will be the screen color. If you want the printing to be black on your inner screen color (the color of the spaces), you need to define the sets with the background color that you used in set 1. Change line 120 above to

```
120 FOR S=1 TO 12
130 CALL COLOR(S,2,16)
140 NEXT S
```

This defines a white background for the first 12 character sets, those sets which have letters and symbols. Now run the program and you will see that the message no longer has the screen color background.

How do you make a simple math drill with graphics?

I have had quite a few requests for an arithmetic drill program. Many readers would like to develop such programs on their own and want to know how to draw a certain number of pictures for the numbers chosen randomly in a simple math problem.

Here is a short program to give you the general idea of using the graphics. I defined character 128 to be the picture. The variables A and B can be numbers from zero to four. Lines 170–200 print the problem on the screen—a simple addition problem. Lines 210 and 220 draw the right number of characters for A and B.

Program 1: Simple Math Drill

```
100 REM  SIMPLE MATH
110 CALL CLEAR
120 CALL CHAR(128,"0024002418")
130 CALL COLOR(13,2,11)
140 RANDOMIZE
150 A=INT(5*RND)
160 B=INT(5*RND)
170 CALL HCHAR(8,10,A+48)
180 CALL HCHAR(10,8,43)
190 CALL HCHAR(10,10,B+48)
200 CALL HCHAR(11,8,95,3)
210 CALL HCHAR(8,12,128,A)
220 CALL HCHAR(10,12,128,B)
230 CALL SOUND(150,1497,4)
240 CALL KEY(0,K,S)
250 IF S<1 THEN 240
260 IF K=32 THEN 400
270 IF K=A+B+48 THEN 310
280 CALL SOUND(100,330,2)
290 CALL SOUND(100,262,2)
300 GOTO 240
310 CALL HCHAR(13,10,K)
320 PRINT "CORRECT!"
330 CALL SOUND(100,262,2)
340 CALL SOUND(100,330,2)
350 CALL SOUND(100,392,2)
360 CALL SOUND(200,532,2)
370 CALL SOUND(1,9999,30)
380 CALL CLEAR
390 GOTO 140
400 CALL CLEAR
410 END
```

If you prefer to have a space between graphics characters, place a character in every other space. You can do this by changing lines 210 and 220 above to the following:

```
210 FOR C=12 TO 12+2*(A-1) STEP2
212 CALL HCHAR(8,C,128)
214 NEXT C
220 FOR C=12 TO 12+2*(B-1) STEP2
222 CALL HCHAR(10,C,128)
224 NEXT C
```

In this sample program, an addition problem is presented and the student answers by pressing a number. If it is incorrect, there is an "uh-oh" sound. If it is correct, an arpeggio is played and the computer goes to the next problem. To stop, press the space bar.

How can you draw a bar graph?

This procedure is similar to the previous sample program. The easiest way to draw a bar graph is to use HCHAR with the appropriate number of repetitions (or VCHAR). You may need to scale the numbers. Take the highest number you'll need to graph, relate it to the greatest number of repetitions you can have in your HCHAR statement, and stay on that row.

Another method is to use PRINT and print the right number of characters for the bar. The following sample program segment demonstrates this method. Character 128 will be a red square. For purposes of illustration, I will use random numbers N up to 90 for the amounts to be graphed. You would probably have specific numbers that have been calculated or read in from DATA.

A is the scaled value (rounded) for N—for every four units one square can be drawn. Line 170 prints the number N then says to start the next printing in the fifth print column. Lines 180–200 print the appropriate number of red squares.

Program 2: Bar Graph Generator

```
100 REM BAR  GRAPH
110 CALL CLEAR
120 CALL COLOR(13,7,7)
130 FOR I=1 TO 10
140 RANDOMIZE
150 N=INT(90*RND)
160 A=INT(N/4+.5)
170 PRINT N;TAB(5);
180 FOR B=1 TO A
190 PRINT CHR$(128);
```

```
200 NEXT B
210 PRINT ::
220 NEXT I
230 GOTO 230
240 END
```

How do you print a list of items in more than two columns?

As you know, the comma in PRINT statements prints items in two columns—items start either in the first print position or the center position. To get three columns or more, use the TAB function. TAB works like the tab key on a typewriter. You may specify which column to start printing. TAB(7) would start the next print item in the seventh print column. Here's a sample that types three columns of names.

```
100 CALL CLEAR
110 READ L$,M$,N$
120 IF L$="@" THEN 180
130 PRINT L$;TAB(10);M$;TAB(19);N$
140 GOTO 110
150 DATA MIKE,BOB,DICK,RICH
160 DATA JIM,JERRY,MARY,PAULA
170 DATA CHRIS,KEVIN,KATHY,KIRK,@,@
,@
180 END
```

How can you print a screen without seeing the scrolling?

Some people don't like to see scrolling as they print. Messages on the TI are always printed on the twenty-fourth row then moved upward. To block this motion, change the screen to black first (because the printing is black), print the messages, then change the screen back to a different color so you can read the printing.

```
100 CALL CLEAR
110 CALL SCREEN(2)
120 PRINT "THIS IS AN EXAMPLE"
130 PRINT ::"TO SEE A SCREEN"
140 PRINT ::"ALL AT ONCE.":::::
150 CALL SCREEN(4)
160 GOTO 160
```

How can you print what is on the screen to the printer?

I'm sorry, but I don't know how to do a *screen dump* of graphics because none of the printers I have right now has the graphics capabilities. You will need to look at your own brand printer manual to see how to use the dot-addressable graphics. If you have a screen of printing, however, with regular printed symbols, you can use the following procedure. The character in each row and column is determined, then that character is printed on the printer. You may need to change the OPEN statement in line 100 to suit your particular printer configuration.

```
100 OPEN #1:"RS232.BA=600"
110 FOR ROW=1 TO 24
120 FOR COL=3 TO 30
130 CALL GCHAR(ROW,COL,G)
140 PRINT #1:CHR$(G);
150 NEXT COL
160 PRINT #1
170 NEXT ROW
180 CLOSE #1
190 END
```

If you want everything you are printing to go both to the screen and to the printer, use both a PRINT statement and a PRINT #1 statement for items printed.

```
100 OPEN #1:"RS232.BA=600"
110 CALL CLEAR
120 PRINT #1:CHR$(12)
130 PRINT "HELLO"
140 PRINT #1:"HELLO"
150 PRINT "ANY MESSAGE"
160 PRINT #1:"ANY MESSAGE"
170 CLOSE #1
180 END
```

Line 120 above goes to the top of a page.

How can you simulate time on the TI?

If you need an exact time, use the CALL SOUND statement in which you can specify an exact duration in milliseconds. If you don't want to hear the sound, use a high frequency and the softest volume.

```
100 PRINT "START"
110 CALL SOUND(1000,9999,30)
120 CALL SOUND(1,9999,30)
130 PRINT "END"
140 END
```

Line 120 is necessary to end the first sound.

If you want to time someone as they are pressing keys to move or are answering a question, use a counter in your CALL KEY loop. You can't relate this counter to an exact time because in each program it will be different—depending on how you do the programming, how long your program is, and how full the memory is. However, once you have your program working, you can print the counter value and use a stopwatch to figure out a formula that relates the actual time to the counter value. ("Type-ette Timer" in my *Programmer's Reference Guide to the TI-99/4A* from COMPUTE! Books uses this technique to time how fast you can type sentences.) Here is a sample:

```
100 T=0
110 CALL KEY(0,K,S)
120 T=T+1
130 IF S<1 THEN 110
140 PRINT T
150 GOTO 100
160 END
```

The faster you press a key, the lower the value for T will be. The longer you wait, the more times the computer will go through the loop and increment T.

Other computers use PRINT AT; how can we do it?

In TI Extended BASIC you can specify a row and column to begin printing an item. However, we don't have that feature in regular console BASIC on the TI. There are several ways to accomplish this, though they're slower than regular printing. First, you can use the regular PRINT statement, perhaps with the TAB function, and then use colons to move the message up to the proper row.

```
PRINT TAB(9);"START PRINTING"::::::
```

The main problem with this method is that it scrolls the screen. If I am labeling graphics, I do all the printing first, then use CALL HCHAR and CALL VCHAR to put up the graphics.

Another method is to treat the letters in the printed message as graphics characters, and use CALL HCHAR to specify the row and column to place the letters on the screen. Here's a general-purpose subroutine that you can use. M$ is the message you want printed, R is the row, and C is the column you want the message to start in.

```
300 FOR L=1 TO LEN(M$)
310 CALL HCHAR(R,C-1+L,ASC(SEG$(M$,
    L,1)))
320 NEXT L
330 RETURN
```

Before you call the subroutine with a GOSUB, specify a row R and a column C and the message M$:

```
900 M$="TEST PRINTING"
910 R=6
920 C=12
930 GOSUB 300
```

How can I put a code in my program?

I have had lots of young people ask me how they can write a program so that whoever runs it must enter a code before the program continues—they don't want their brothers and sisters using their program. The general idea is that you put a code name in the program as a string variable. Next, use INPUT for the user who is running the program to type in the code. Now compare the INPUT value with the code to see whether to continue or not.

```
100 CALL CLEAR
110 CODE$="RANDY"
120 INPUT "ENTER CODE NAME: ":A$
130 IF A$=CODE$ THEN 160
140 PRINT ::"SORRY, INVALID CODE."
150 STOP
160 REM PROGRAM CONTINUES
```

The only problem with specifying the code in line 110 is that anyone can load the program, then LIST it to find out what the code name is. One method I use so people can't read the code name is to hold down the CTRL key (key with the red dot) while you type your code message. Line 110 will now look like this:

```
110 CODE$="     "
```

or you may get some funny-looking graphics characters between the quotes. Now when someone lists your program, they can't tell what the code name is. When *you* run the program, be sure to hold the CTRL key down when you INPUT the code name, and it will match the code in the program.

A Couple Of Warnings

Always use the SHIFT key on the left side of the keyboard to type the plus sign. You don't want to go for the right SHIFT key and accidentally hit the FCTN key—and *quit*!

Do *not* use TI Extended BASIC to run regular TI BASIC programs because they may not run properly. One reason is the double colon used in PRINT statements, and another reason is that I often use graphics in character sets 15 and 16, which are not available in Extended BASIC.

If you have a disk drive attached to your computer, the disk uses up some memory. For any of my published programs, type in CALL FILES(1) and press ENTER, then type NEW and press ENTER, then proceed normally (load a program or start typing a program). This procedure clears about 1000 bytes of memory so a program can fit.

Until Next Time ...

I hope these ideas help you in your programming. Your computer can be a lot of fun. Part of the joy of programming is getting that machine to do what you want it to do. As I continue these columns I hope to present a variety of programs so you can see that this computer is really quite versatile. Your suggestions and letters are always welcome. ©

Programming 64 Sound
Part 2

John Michael Lane

Last month in Part 1, we discussed sound and music in general. This month we examine some techniques for programming more complicated music using the 64's SID chip.

The control register is the most complex register in the chip. *Each* of the eight bits in this register has a different function. Dealing with individual bits within a one-byte register is often a problem for BASIC programmers. One very easy way to approach the problem is to use the following:

```
170 B(0)=1
180 B(1)=0
190 B(2)=1
200 B(3)=0
210 B(4)=0
220 B(5)=0
230 B(6)=0
240 B(7)=1
250 FOR I=0 TO 7
260 Q=Q+B(I)*2↑I
270 NEXT I:POKE S+4,Q
```

This is not efficient programming, but by defining the bits we want (that is, B(I) where I= the bit number) in terms of a 1 and those we don't want in terms of a 0, this segment will work. It will be somewhat slow and cannot be used in a loop that must execute quickly, which is usually the case when doing musical programming.

A quicker method is to think of the bits in terms of their value in an eight-bit binary number. Bit 0 has a value of 1, bit 1=2, bit 2=4, bit 3=8, bit 4=16, bit 5=32, bit 6=64, and bit 7=128. In the case above, we want to set bits 0, 2, and 7 *on*, so we simply add their values: 1+4+128=133. Simply POKE 133 into the register to set those bits. It's much simpler, but requires you to add up the bit values before writing the program, so when you look back on the program one month later you may not have the slightest idea why you chose 133.

The first bit of the control register, bit 0, acts as the gate to turn the sound on and off. Remember that when the gate is opened (when bit 0 is set to 1), the attack phase of the volume envelope begins. When the gate is closed (bit 0 is set to 0), the release phase of the volume envelope is triggered. If the gate is closed prematurely, the sustain, decay, and even a portion of the attack phase may be omitted. Opening and closing the gate is actually very easy. Just remember that POKEing an odd value in register 4 turns the gate on and that POKEing an even value into the register turns the gate off.

Watch The Timing

Be careful of turning the gate off by POKEing zero into the register. That will also clear the waveform bits (which we'll discuss in a second) and will result in your volume envelope having no release phase.

The next bit, bit 1, is the *sync* bit. If this bit is on, the output from voice 1 will be synchronized with the output from voice 3. *Sync* in this case means that the output of voice 1 will be replaced with a logical AND of the output of voice 1 and voice 3. Another way to think of it is that voice 1 is turned on and off with the frequency of voice 3. In order for this bit to have any effect, oscillator three (voice 3) must be set to some frequency less than voice 1. The best way to understand this effect is to listen to it. "Laser" (Program 4) contains a demonstration using the sync bit. When using sync, the lower frequency will predominate. The effect works best when the lower frequency is 1/10 to 1/2 of the higher.

The sync bit has a slightly different effect in the other two voices. In voice 2 it produces a sync of voice 2 with voice 1, and in voice 3 it produces a sync of voice 3 with voice 2.

The next bit, bit 2, is the *ring modulation* bit. When this bit is set on, it produces nonharmonic overtones that sound like a bell. In order for this effect to take place, the triangular waveform must be selected for voice 1, and voice 3 must have a frequency other than zero.

Ring modulation in the other voices works like the sync bit; that is, for voice 2 to be ring modulated, voice 1 must have a nonzero frequency. For voice 3, voice 2 must be nonzero. In all cases the triangular waveform must be selected for the affected voice.

Bit 3 in the control register is the test bit. Setting the test bit to one will turn off the sound generator. This technique will generally be used only by machine language programmers.

Bits 4–7 are the waveform bits. Turning on bit 4 will select the triangular waveform; bit 5 will select the sawtooth; bit 6, the rectangular pulse; and bit 7, white noise (the hissing sound that you hear between stations on a radio).

At this point you must be asking yourself "What happens if more than one bit is selected?" The answer is that the two (or more) waveforms will be ANDed together (a logical AND will be done on the waveforms). Commodore cautions that selecting more than one waveform while using the white noise waveform could cause the oscillator to go silent, so don't combine waveforms using the white noise waveform. Even while avoiding the white noise waveform, it is still possible to generate four more waveform shapes using combinations of the sawtooth, triangular, and rectangular pulse waveforms. However, the volume declines significantly when combining waveforms.

Register 5 contains the attack and decay values for the sound envelope. The four-bit attack value is held in bits 7–4. The four-bit decay value is held in bits 3–0. The values can be loaded like this:

```
300 A=13:D=5: REM ATTACK=13,DECAY=5
310 POKE S+5,16*A+D
```

Register 6 contains the sustain level and the release value. As above, the sustain level is held in bits 7–4, and the release value in bits 3–0. Program them in the following manner:

```
320 SU=13:R=4: REM SUSTAIN=13,RELEASE=4
330 POKE S+6,16*SU+R
```

Now we've completely covered the seven register groups and shown how to load them. "Twiddle" (Program 1) allows you to explore all possible combinations using these seven registers. The program allows you to set and change any of the values and then listen to an eight-note scale governed by those values. If you can sit down and play with the program for a couple of hours, you'll gain a good understanding of how changing SID parameters affects a sound. The program is also useful for demonstrating how to play a tune within a basic program.

From Sound To Music

To play actual music, you generally write a program which will load all the parameters except the waveform and the frequency. At this point you select the note to be played and POKE the appropriate values into the frequency registers. Then you POKE the waveform value plus one (16+1=17 for triangular, 33 for sawtooth, 65 for the rectangular pulse, and 129 for white noise) into register 4 (the control register). Adding a 1 causes the gate bit (bit 0) to be turned on and the tone begins. The program waits a certain period of time and then POKEs the waveform value (16, 32, 64, or 128) into register 4. By POKEing an even number into the register we turn the gate off, and the note begins its release phase and gradually dies out (according to the release value that you've set).

A simple way to time the note is to use a delay loop. An empty loop (like the one below) will execute 1000 cycles in just about one second.

```
400 FOR I=1 TO 1000:NEXT I
```

Therefore, each cycle is just about 1/1000 second (or a millisecond). To turn the note on and off, the program line will look like this:

```
400 POKE S+4,17:FOR I=1 TO 250:
    NEXT:POKES+4,16
```

The above program line will play a note for about one quarter of a second.

This technique works well for a single voice, but it may not work at all for more than one voice. The problem is that while the computer is timing the duration of one note, it cannot be separately timing voices 2 and 3. We could fill the empty loop with timing routines for voices 2 and 3, but that would change the execution time for the loop and throw the timing off.

A second technique is to use the internal timer of the Commodore 64 through the use of the variable TI. The variable TI is updated automatically on the Commodore 64 and increases by a value of one every 1/60 second. We can use this timer to time the duration of our notes:

```
500 T0=TI: REM INITIALIZE THE VARIABLE "T
    0"
510 T0=T0+D: REM INCREASE "T0" BY DURATIO
    N OF THE FIRST NOTE - D
520 IF T0<=TI THEN GOSUB 1100: REM CHECK
    {SPACE}IF THE TIME IS UP
525 REM IF SO SUBROUTINE 1100 WILL CHANGE
     NOTES
530 GOTO 520: REM IF NOT CHECK TIME AGAIN
```

The key to using this routine is to make sure that the subroutine executes quickly, at least while using multiple voices. "Tune" (Program 2) illustrates this technique using all three voices. But this method isn't problem-free either. We want to reproduce the rhythm of the original tune as accurately as possible. It's physically impossible to change the frequency of all three voices at once. Using BASIC, it's somewhat difficult to change all three voices in less than 1/6 second. For that reason, we split all the frequencies into the higher and lower order bytes before the tune begins. We can then change the frequency of all three voices in about 1/10 second. For most tunes that will be satisfactory. However, for a fast tempo, you might have to omit the second or third voice in order to maintain the rapid changes of the first voice.

Sound Effects

Now, let's briefly explore the sound of a laser firing, or an explosion, siren, or any other sound we need. How can we accomplish it?

There is no direct way. The best approach is trial and error. Listen to the sound carefully. Most sounds in nature cannot be duplicated simply by selecting the right waveform and envelope. Generally, the frequency is also actively changing during the sound's life. While you listen to (or think about) the sound you want, consider what is happening to the frequency. Is it rising or falling? How quickly?

Also consider the volume. Many volume envelopes cannot be duplicated using the attack/decay/sustain/release envelope on the Commodore SID. You will often have to change the volume level through program control, using the volume register (register 24) on the SID.

Programs 3 and 4, "Blast-off" and "Laser," illustrate one approach. In Blast-off, both the frequency and volume are modulated by the program. Laser demonstrates the sync feature and modulates the frequency to produce the laser sound. Both programs were written after much trial and error.

Many authors, when converting programs to the 64, simply drop the sound effects or stop at a sound which is only vaguely like the one they want. Be persistent; the 64 can accurately produce almost any sound. As you gain experience, you will find that the trial and error phase will decrease significantly.

Twiddle illustrates the basic methods of loading the SID registers and lets you experiment by changing the waveform and ADSR envelope while listening to the musical scale.

Tune uses the three voices to play an English folk tune. Don't be discouraged by the long list of DATA statements. Voice 1 repeats the same statements four times, and there is considerable repetition in voices 2 and 3. Once you've typed in the few basic lines, you can simply change the line numbers with the screen editor to produce the remainder of the data statements.

Tune can be used to produce any melody by changing the values in the DATA statements. Each note is represented by a pair of values. The first represents the duration of the note (in sixtieths of a second). A value of 30–40 is appropriate for a quarter note. The second value is the frequency of the note. Appendix E in the *Commodore 64 Programmer's Reference Guide* offers a good, simple frequency table. Below are the values for the 12-semitone scale starting at middle C.

C - 4291	C# - 4547
D - 4817	D# - 5103
E - 5407	
F - 5728	F# - 6069
G - 6430	G# - 6812
A - 7217	A# - 7647
B - 8101	

Notes for other octaves can be calculated by doubling or halving these values, depending upon whether you're going one octave up (doubling) or one octave down (halving).

It is useful to convert one measure of music to one DATA statement if you can. This makes it easier to match the voices.

Voice 1 is the sound of a flute, voice 2 is a mandolin, and voice 3 is a guitar. Blast-off and Laser are supposed to produce the sound of their titles. They are pretty straightforward.

Program 1: Twiddle

Refer to the "Automatic Proofreader" article before typing this program in.

```
5 S=54272                                   :rem 201
7 DIM A(15),D(15)                            :rem 48
10 FORL=STOS+24:POKEL,0:NEXT                 :rem 53
15 GOSUB 1000                               :rem 167
17 GOSUB 1100                               :rem 170
18 GOSUB 1200                               :rem 172
20 PRINT"{CLR}";TAB(5);"TOUCH W FOR WAVEF
   ORM"                                        :rem 5
30 PRINT TAB(5)"TOUCH A FOR ATTACK RATE"
                                             :rem 32
40 PRINT TAB(5)"TOUCH S FOR SUSTAIN LEVEL
   "                                         :rem 238
45 PRINT TAB(5)"TOUCH T FOR SUSTAIN TIME"
                                             :rem 171
50 PRINT TAB(5)"TOUCH R FOR RELEASE"
                                              :rem 80
60 PRINT TAB(5)"TOUCH D FOR DECAY"
                                             :rem 168
70 PRINT TAB(5)"TOUCH P FOR PULSE WIDTH"
                                              :rem 88
72 PRINT TAB(5)"TOUCH B TO SET DEAD TIME"
                                              :rem 40
75 PRINT TAB(5)"TOUCH + OR - FOR FREQUENC
   Y CHANGE"                                  :rem 85
80 GET A$:IF A$=""THEN80                     :rem 243
82 IF A$="W"THEN 200                         :rem 247
```

```
84 IF A$="A" THEN 250                :rem 232
86 IF A$="S" THEN 300                :rem 248
88 IF A$="R" THEN 350                :rem 254
90 IF A$="D" THEN 400                :rem 229
92 IF A$="P" THEN 450                :rem 248
94 IF A$="T" THEN 500                :rem 250
96 IF A$="+" THEN GOSUB 1400         :rem 131
97 IF A$="B" THEN 550                :rem 240
98 IF A$="-" THEN GOSUB 1450         :rem 140
100 REM                              :rem 117
105 POKE S+24,15                     :rem 59
110 POKE S+5,16*A+D                  :rem 225
120 POKE S+6,16*SL+R                 :rem 79
130 POKE S+3,INT(P/256)              :rem 248
140 POKE S+2,P-256*INT(P/256)        :rem 60
150 FOR I=1 TO 8                     :rem 15
160 IFINT(F(I))<=65536THENPOKE S+1,INT(F(
    I)/256)                          :rem 229
170 POKE S,F(I)-256*INT(F(I)/256)    :rem 2
180 IFINT(F(I))<=65536THENPOKE S+4,2↑(W+3
    )+1                              :rem 244
185 FORJ=1TOT:NEXT                   :rem 173
187 POKE S+4,2↑(W+3)                 :rem 67
188 FORJ=1TOB:NEXT                   :rem 158
190 NEXT I:GOTO 20                   :rem 247
200 PRINT"WAVEFORM IS";" - ";W       :rem 164
202 PRINT"1=TRIANGLE"                :rem 41
204 PRINT"2=SAWTOOTH"                :rem 79
206 PRINT"3=PULSE"                   :rem 98
208 PRINT"4=NOISE"                   :rem 90
210 INPUT"ENTER WAVEFORM (1-4)";W:rem 193
215 IFW<1 ORW>4THEN210               :rem 23
220 GOTO 100                         :rem 94
250 PRINT"ATTACK RATE IS";A          :rem 100
260 INPUT"ENTER ATTACK RATE (0-15)";A
                                     :rem 94
265 IFA<0ORA>15THEN260               :rem 38
270 GOTO 100                         :rem 99
300 PRINT"SUSTAIN LEVEL IS";SL       :rem 121
310 INPUT"ENTER SUSTAIN LEVEL (0-15)";SL
                                     :rem 115
315 IFSL<0ORSL>15THEN310             :rem 218
320 GOTO 100                         :rem 95
350 PRINT"RELEASE RATE IS";R         :rem 191
360 INPUT"ENTER RELEASE RATE (0-15)";R
                                     :rem 185
365 IFR<0ORR>15THEN360               :rem 74
370 GOTO 100                         :rem 100
400 PRINT"DECAY RATE IS";D           :rem 18
410 INPUT"ENTER DECAY RATE (0-15)";D
                                     :rem 12
415 IFD<0ORD>15THEN410               :rem 38
420 GOTO 100                         :rem 96
450 PRINT"PULSE WIDTH IS";100*P/4095
                                     :rem 86
460 INPUT"ENTER PULSE WIDTH (0-100)";P
                                     :rem 191
465 IFP<0ORP>100THEN460              :rem 115
470 P=P*4095/100                     :rem 52
480 GOTO 100                         :rem 102
500 PRINT"SUSTAIN TIME IS";T;"MILLISECOND
    S"                               :rem 236
510 PRINT"MINIMUM TIME FOR ATTACK/DECAY C
    YCLE IS:"                        :rem 44
515 PRINT A(A)+D(D);"MILLISECONDS" :rem 4
520 INPUT"ENTER TIME IN MILLISECONDS";T
                                     :rem 196
530 GOTO 100                         :rem 98
550 PRINT"DEAD TIME IS";B;"MILLISECONDS"
                                     :rem 198
560 INPUT"INPUT DEAD TIME IN MILLISECONDS
    ";B                              :rem 214
570 GOTO 100                         :rem 102
1000 W=1:A=8:D=6:R=9:SL=12:P=2000:T=302
                                     :rem 203
1010 RETURN                          :rem 162
1100 FORI=1TO8:READF(I):NEXT         :rem 234
1110 DATA 4291,4817,5407,5728,6430,7217,8
     101,8538                        :rem 155
1120 RETURN                          :rem 164
1200 FOR I=0TO15:READ A(I):D(I)=3*A(I):NE
     XT                              :rem 160
1210 DATA 2,8,16,24,38,56,68,80,100,250,5
     00,800,1000,3000,5000,7000     :rem 186
1220 RETURN                          :rem 165
1400 FOR I=1TO 8:F(I)=F(I)*2:NEXT:RETURN
                                     :rem 100
1450 FOR I=1TO8:F(I)=F(I)/2:NEXT:RETURN
                                     :rem 110
```

Program 2: Tune

Refer to the "Automatic Proofreader" article before typing this program in.

```
5 DIM D(3,200),F(3,200),G(3,200) :rem 254
10 S=54272                           :rem 245
20 FORI=0TO24:POKES+I,0:NEXT         :rem 13
30 FORI=1TO3                         :rem 215
40 J=1                               :rem 28
50 READ D(I,J),F(I,J):REM GET FREQ & DURA
   TION                              :rem 15
55 G(I,J)=INT(F(I,J)/256):F(I,J)=F(I,J)-2
   56*G(I,J)                         :rem 202
60 IF F(I,J)=0 AND D(I,J)=0 THEN 90
                                     :rem 228
70 J=J+1:GOTO 50                     :rem 108
90 PRINT "VOICE";I;" ";J;" NOTES" :rem 64
100 NEXT I                           :rem 25
110 POKES+24,15                      :rem 55
200 REM SET VOICE ONE                :rem 186
210 W1=16:REM TRIANGLE WAVEFORM      :rem 154
220 POKES+5,6*16+0:REM ATTACK=6,DECAY=0
                                     :rem 12
230 POKES+6,10*16+0:REM SUSTAIN=10,RELEAS
    E=0                              :rem 110
300 REM SET VOICE TWO                :rem 211
310 W2=32:REM SAWTOOTH WAVEFORM      :rem 189
320 POKES+12,0*16+9:REM ATTACK=0,DECAY=9
                                     :rem 65
330 POKES+13,00*16+0:REM SUSTAIN=00,RELEA
    SE=00                            :rem 203
400 REM SET VOICE THREE              :rem 82
410 W3=64:REM RECTANGULAR WAVE       :rem 79
420 POKES+17,3:REM DUTY CYCLE 20%:rem 101
430 POKES+19,3*16+10:REM ATTACK=3,DECAY=1
    0                                :rem 160
440 POKES+20,0*16+0:REM SUSTAIN=0:RELEASE
    =O                               :rem 104
500 J=0:K=0:L=0:T1=TI:T2=T1:T3=T1:rem 207
600 IF T1=<TI THEN GOSUB 1100        :rem 49
610 IF T2=<TI THEN GOSUB 1200        :rem 52
620 IF T3=<TI THEN GOSUB 1300        :rem 55
630 GOTO 600                         :rem 104
1000 ON I GOTO 1100,1200,1300        :rem 129
1100 J=J+1:T1=T1+D(1,J)              :rem 215
1115 IFD(1,J)=0 THEN POKES+4,W1:POKES+11,
     W2:POKES+18,W3:END              :rem 217
1117 POKES+4,W1                      :rem 95
1120 POKES,F(1,J):POKES+1,G(1,J)  :rem 51
1140 POKES+4,W1+1:RETURN             :rem 209
1200 K=K+1:T2=T2+D(2,K)              :rem 222
```

```
1210 POKE S+11,W2                          :rem 136
1220 POKE S+7,F(2,K):POKES+8,G(2,K)
                                           :rem 161
1240 POKES+11,W2+1:RETURN                    :rem 1
1300 L=L+1:T3=T3+D(3,L)                    :rem 229
1310 POKES+18,W3                           :rem 145
1320 POKES+14,F(3,L):POKES+15,G(3,L)
                                             :rem 2
1340 POKES+18,W3+1:RETURN                   :rem 10
2000 REM NOTES FOR VOICE ONE               :rem 110
2010 DATA 30,4051                           :rem 54
2020 DATA 30,5407,30,4051,30,6069,30,4051
                                           :rem 215
2030 DATA 30,6430,30,6069,30,5407,30,4050
                                           :rem 218
2040 DATA 30,5407,30,4050,30,6069,30,4050
                                           :rem 215
2050 DATA30,6430,30,7217,30,8101,30,4050
                                           :rem 210
2060 DATA30,5407,30,4050,30,6069,30,4050
                                           :rem 217
2070 DATA30,6430,30,6069,30,5407,30,4050
                                           :rem 222
2080 DATA30,5407,30,4050,30,6069,30,4817
                                           :rem 230
2090 DATA60,5407,30,5407,30,4050  :rem 86
2120 DATA 30,5407,30,4051,30,6069,30,4051
                                           :rem 216
2130 DATA 30,6430,30,6069,30,5407,30,4050
                                           :rem 219
2140 DATA 30,5407,30,4050,30,6069,30,4050
                                           :rem 216
2150 DATA30,6430,30,7217,30,8101,30,4050
                                           :rem 211
2160 DATA 30,5407,30,4050,30,6069,30,4050
                                           :rem 218
2170 DATA30,6430,30,6069,30,5407,30,4050
                                           :rem 223
2180 DATA30,5407,30,4050,30,6069,30,4817
                                           :rem 231
2190 DATA120,5407                          :rem 117
2220 DATA 30,5407,30,4051,30,6069,30,4051
                                           :rem 217
2230 DATA 30,6430,30,6069,30,5407,30,4050
                                           :rem 220
2240 DATA 30,5407,30,4050,30,6069,30,4050
                                           :rem 217
2250 DATA30,6430,30,7217,30,8101,30,4050
                                           :rem 212
2260 DATA30,5407,30,4050,30,6069,30,4050
                                           :rem 219
2270 DATA30,6430,30,6069,30,5407,30,4050
                                           :rem 224
2280 DATA30,5407,30,4050,30,6069,30,4817
                                           :rem 232
2290 DATA120,5407                          :rem 118
2320 DATA 30,5407,30,4051,30,6069,30,4051
                                           :rem 218
2330 DATA 30,6430,30,6069,30,5407,30,4050
                                           :rem 221
2340 DATA 30,5407,30,4050,30,6069,30,4050
                                           :rem 218
2350 DATA30,6430,30,7217,30,8101,30,4050
                                           :rem 213
2360 DATA30,5407,30,4050,30,6069,30,4050
                                           :rem 220
2370 DATA30,6430,30,6069,30,5407,30,4050
                                           :rem 225
2380 DATA30,5407,30,4050,30,6069,30,4817
                                           :rem 233
2390 DATA120,5407                          :rem 119
2900 DATA 0,0                              :rem 113
3000 REM NOTES FOR VOICE TWO               :rem 135
3010 DATA990,0                             :rem 220
3020 DATA60,2703,60,2408                   :rem 201
3030 DATA30,2145,30,2025,60,2145            :rem 73
3040 DATA60,2025,60,1804                   :rem 199
3050 DATA30,1607,30,1517,60,1351            :rem 80
3060 DATA60,2703,60,2408                   :rem 205
3070 DATA30,2145,30,2025,60,2145            :rem 77
3080 DATA60,2025,60,1804                   :rem 203
3090 DATA30,1607,30,1517,60,1351            :rem 84
3120 DATA60,2703,60,2408                   :rem 202
3130 DATA30,2145,30,2025,60,2145            :rem 74
3140 DATA60,2025,60,1804                   :rem 200
3150 DATA30,1607,30,1517,60,1351            :rem 81
3160 DATA60,2703,60,2408                   :rem 206
3170 DATA30,2145,30,2025,60,2145            :rem 78
3180 DATA60,2025,60,1804                   :rem 204
3190 DATA30,1607,30,1517,60,1351            :rem 85
3220 DATA60,2703,60,2408                   :rem 203
3230 DATA30,2145,30,2025,60,2145            :rem 75
3240 DATA60,2025,60,1804                   :rem 201
3250 DATA30,1607,30,1517,60,1351            :rem 82
3260 DATA60,2703,60,2408                   :rem 207
3270 DATA30,2145,30,2025,60,2145            :rem 79
3280 DATA60,2025,60,1804                   :rem 205
3290 DATA30,1607,30,1517,60,1351            :rem 86
3900 DATA 0,0                              :rem 114
4000 REM NOTES FOR VOICE THREE               :rem 6
4010 DATA1950,0                             :rem 10
4020 DATA 60,2703,60,2408                  :rem 202
4030 DATA 30,2703,15,2703,15,2703,60,2025
                                           :rem 215
4040 DATA 30,2703,30,2703,30,3034,30,3034
                                           :rem 206
4050 DATA 15,3215,15,3215,15,3215,15,3215
,60,3034                                    :rem 99
4060 DATA 45,4050,15,3608,45,4050,15,3608
                                           :rem 234
4070 DATA 45,4050,15,3608,15,4050,15,3608
,15,3215,15,3034                           :rem 249
4080 DATA 60,2703,60,2408                  :rem 208
4090 DATA 30,2703,15,2703,15,2703,60,2025
                                           :rem 221
4100 DATA 30,2703,30,2703,30,3034,30,3034
                                           :rem 203
4110 DATA 15,3215,15,3215,15,3215,15,3215
,60,3034                                    :rem 96
4120 DATA 45,4050,15,3608,45,4050,15,3608
                                           :rem 231
4130 DATA 45,4050,15,3608,15,4050,15,3608
,15,3215,15,3034                           :rem 246
4140 DATA 60,2703,60,2408                  :rem 205
4150 DATA 30,2703,15,2703,15,2703,60,2025
                                           :rem 218
4160 DATA 60,4050,60,4050                  :rem 199
4170 DATA 30,4050,15,4050,15,4050,60,4050
                                           :rem 211
4900 DATA 800,0,0,0                        :rem 147
```

Program 3: Blast-off

Refer to the "Automatic Proofreader" article before typing this program in.

```
10 S=54272                                 :rem 245
20 FOR I=STOS+24:POKEI,0:NEXT               :rem 48
30 POKES+24,15                               :rem 8
40 FR=0500                                 :rem 254
```

```
50 A=Ø:D=Ø:SS=15:R=Ø              :rem 122
60 W=128:P=1024                    :rem 35
70 POKES+1,INT(FR/256)             :rem 17
80 POKES,FR-256*INT(FR/256)        :rem 66
90 POKES+3,INT(P/256)             :rem 205
100 POKES+2,P-256*INT(P/256)       :rem 56
110 POKES+5,16*A+D                :rem 225
120 POKES+6,16*SS+R                :rem 86
200 POKES+4,W+1:REM TURN SOUND ON:rem 223
210 FORI=200TO1 STEP-1               :rem Ø
220 FR=FR+100:REM INCREASE FREQUENCY
                                  :rem 215
222 IF I< 45 THEN POKES+24,I/3:REM NEAR T
    HE END TURN DOWN THE VOLUME    :rem 98
225 F2=INT(FR/256):F1=FR-256*F2   :rem 224
230 POKES,F1:POKES+1,F2           :rem 118
240 NEXT I                         :rem 30
250 POKES+4,W:REM TURN SOUND OFF :rem 198
```

Program 4: Laser

Refer to the "Automatic Proofreader" article before typing this program in.

```
10 S=54272                        :rem 245
20 FOR I=STOS+24:POKEI,Ø:NEXT      :rem 48
30 POKES+24,143                    :rem 58
40 FR=50000                        :rem 46
50 A=Ø:D=8:SS=15:R=Ø8             :rem 186
60 W=Ø64:P=1024                    :rem 34
70 POKES+1,INT(FR/256)             :rem 17
80 POKES,FR-256*INT(FR/256)        :rem 66
90 POKES+3,INT(P/256)             :rem 205
100 POKES+2,P-256*INT(P/256)       :rem 56
110 POKES+5,16*A+D                :rem 225
120 POKES+6,16*SS+R                :rem 86
130 POKES+15,75                    :rem 63
155 POKES+4,W+3:REM USING W+3 TURNS ON
    {2 SPACES}GATE AND SYNC        :rem 32
160 FORI=1TO25                     :rem 63
170 POKES+15,120-4*I:REM{2 SPACES}DECREAS
    E FREQ VOICE THREE            :rem 180
180 NEXT I                         :rem 33
185 POKES+4,W                       :rem 2
```

Applesoft Lister

David Dobrin

"Applesoft Lister" will give you more readable program listings, along with printer-oriented output, translated control characters, and indention of nested FOR-NEXT loops.

Would you like your Applesoft programs to look like this:

```
10 REM BASIC LISTING WITH APPLESOFT LIST
20 HOME
22 PRINT "ANT SCRAM[G][G][G]"
30 FOR J=0 TO 35
31    VTAB 2
  :    HTAB J+1
40    PRINT " ;=;@"
50    NEXT
60 PRINT "[G][G][G]THAT IS ALL"
```

instead of this:

```
10   REM  BASIC LISTING WITH APPLESOFT LIST
20   HOME
22   PRINT "ANT SCRAM"
30   FOR J = 0 TO 35
31   VTAB 2: HTAB J + 1
40   PRINT " ;=;@"
50   NEXT
60   PRINT "THAT IS ALL"
```

Applesoft programs are usually very difficult to read. The standard LIST function built into Applesoft is unsophisticated, having only the minimum logic necessary to list programs. Here's a program for the Apple that will list Applesoft programs in a nicely formatted fashion. Five major features distinguish "Applesoft Lister" from the standard format:

- There is intelligent spacing between keywords, variables, and operands.
- Multiple statements with a single line number are listed one per line.
- FOR-NEXT constructs are nested.
- Output is oriented for a printer. This listing will not simply "wrap" when it runs out of space on a line.
- Control characters are shown with printable characters.

How Applesoft Lister Works

The program translates the Applesoft intermediate language (IL) into statement numbers and keywords. The keywords are taken from ROM at $D0D0. If this program is to be used with Applesoft in RAM, this value must be changed.

The high byte of the keyword table address is at location $812C. The low byte is at $8130.

When a colon (:) is encountered in the text, the lister starts a new line, indenting appropriately. No action is taken on colons inside double quotes or REM statements. FORs and NEXTs are observed to calculate a nest level.

If you would like to change the indentation of your FOR-NEXT constructs or multiple statements you can change the value at location 32771 with the POKE command. Putting a 0 there will turn indenting off, a 3 will indent three spaces per nest level, a 10 will indent ten spaces per nest level, and so on.

If you want to change the column width, change the value at 32772 with the POKE command. Putting a 39 there will give a screen width. You can also use 80, 132, or whatever your printer width is.

These POKEs can, of course, be made permanent by saving the program to disk or tape after changing.

Control characters are printed inside brackets; for example, CTRL-G appears as [G].

Loading The Program Into Your Apple

The lister program is written entirely in machine language. Program 1 is a BASIC program which READs the machine language from DATA statements and POKEs it into memory.

The program was assembled to load at location $8000. If your machine has less than 48K, the program will have to be relocated.

If you wish to enter the machine language, you can do so by using the Apple monitor (CALL –151). Enter the hex values as shown in Program 2. The *Apple Reference Manual*, Chapter 3, details the use of the resident monitor.

Once the program is entered into the Apple either by the loader or from the monitor, it should be saved to disk or tape before going any farther. This can be done by typing:

]BSAVE ALIST,A$8000,L$2F0

or

***8000.82F0W**

Running Applesoft Lister

After the program has been stored, it can be utilized by loading the Applesoft program to be listed in the usual manner. The list program can then be loaded with:

]BLOAD ALIST

or

]CALL −155
***8000.82F0R**
***(CTRL-C)**

The listing program can then be run by typing:

]PR#*x* (where *x* is the slot for your printer interface, if you want the output to go to a printer)
]CALL 32768

Program 1: BASIC Loader For Applesoft Lister

```
100  FOR I = 32768 TO 33295
110  READ A:CK = CK + A: POKE I,A
120  NEXT
130  IF CK <  > 47880 THEN  PRINT "ERRO
     R IN DATA STATEMENTS": STOP
140  PRINT "LISTER ML LOADED"
150  END
200  DATA  76,5,128,3,80,169,Ø,133
210  DATA  10,169,1,133,Ø,169,8,133
220  DATA  1,169,141,32,157,129,32,96
230  DATA  129,133,2,32,96,129,133,3
240  DATA  5,2,208,1,96,32,96,129
250  DATA  133,4,32,96,129,133,5,169
260  DATA  Ø,133,6,133,7,133,8,162
270  DATA  16,24,248,165,6,101,6,133
280  DATA  6,165,7,101,7,133,7,165
290  DATA  8,101,8,133,8,216,6,4
300  DATA  38,5,144,2,230,6,202,208
310  DATA  224,162,5,160,Ø,165,8,41
320  DATA  15,208,12,192,Ø,208,8,224
330  DATA  1,240,4,169,160,208,4,160
340  DATA  1,9,176,32,157,129,152,72
350  DATA  160,4,6,6,38,7,38,8
360  DATA  136,208,247,104,168,202,208,
     213
370  DATA  169,160,32,157,129,32,107,12
     9
380  DATA  169,Ø,133,9,32,96,129,201
390  DATA  Ø,208,16,169,141,32,157,129
400  DATA  165,2,133,Ø,165,3,133,1
410  DATA  76,22,128,166,14,236,4,128
420  DATA  48,18,72,162,Ø,189,149,129
430  DATA  240,6,32,157,129,232,208,245
440  DATA  32,107,129,104,201,34,208,8
450  DATA  165,9,73,128,133,9,169,34
460  DATA  166,9,208,27,201,58,208,19
470  DATA  162,Ø,189,141,129,240,6,32
480  DATA  157,129,232,208,245,32,107,1
     29
490  DATA  76,148,128,201,128,16,26,41
500  DATA  127,201,32,16,14,72,169,91
510  DATA  32,157,129,104,9,64,32,157
520  DATA  129,169,93,32,157,129,76,148
530  DATA  128,72,201,129,208,2,230,10
540  DATA  201,130,208,2,198,10,201,178
550  DATA  208,2,230,9,170,188,37,129
560  DATA  132,11,36,11,16,5,169,160
570  DATA  32,157,129,169,208,133,13,16
     9
580  DATA  208,133,12,104,170,160,Ø,202
590  DATA  16,16,177,12,230,12,208,2
600  DATA  230,13,201,128,16,241,48,242
610  DATA  160,Ø,177,12,200,170,32,157
620  DATA  129,138,16,246,36,11,80,5
630  DATA  169,160,32,157,129,76,148,12
     8
640  DATA  160,Ø,177,Ø,230,Ø,208,2
650  DATA  230,1,96,162,13,134,14,166
660  DATA  10,16,2,162,Ø,224,6,48
670  DATA  2,162,6,202,48,14,172,3
680  DATA  128,136,48,247,169,160,32,15
     7
690  DATA  129,76,129,129,96,141,160,16
     Ø
700  DATA  160,160,160,186,Ø,141,160,16
     Ø
710  DATA  160,160,160,160,Ø,9,128,32
720  DATA  237,253,230,14,96,64,64,64
730  DATA  64,64,64,64,64,64,64,Ø
740  DATA  Ø,64,64,64,64,64,64,Ø
750  DATA  64,64,64,64,64,Ø,Ø,64
760  DATA  64,64,64,64,64,Ø,64,64
770  DATA  Ø,Ø,64,64,64,64,Ø,64
780  DATA  64,64,64,64,Ø,64,64,Ø
790  DATA  64,64,64,64,64,64,64,64
800  DATA  64,64,64,64,64,Ø,192,Ø
810  DATA  Ø,192,192,64,192,Ø,Ø,Ø
820  DATA  Ø,Ø,192,192,Ø,Ø,Ø,Ø
830  DATA  Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø
840  DATA  Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø
850  DATA  Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø
```

Program 2: Hex Dump Of Applesoft Lister Machine Language

```
8000- 4C 05 80 03 50 A9 00 85
8008- 0A A9 01 85 00 A9 08 85
8010- 01 A9 8D 20 9D 81 20 60
8018- 81 85 02 20 60 81 85 03
8020- 05 02 D0 01 60 20 60 81
8028- 85 04 20 60 81 85 05 A9
8030- 00 85 06 85 07 85 08 A2
8038- 10 18 F8 A5 06 65 06 85
8040- 06 A5 07 65 07 85 07 A5
8048- 08 65 08 85 08 D8 06 04
8050- 26 05 90 02 E6 06 CA D0
8058- E0 A2 05 A0 00 A5 08 29
8060- 0F D0 0C C0 00 D0 08 E0
8068- 01 F0 04 A9 A0 D0 04 A0
8070- 01 09 B0 20 9D 81 98 48
8078- A0 04 06 06 26 07 26 08
8080- 88 D0 F7 68 A8 CA D0 D5
8088- A9 A0 20 9D 81 20 6B 81
8090- A9 00 85 09 20 60 81 C9
8098- 00 D0 10 A9 8D 20 9D 81
80A0- A5 02 85 00 A5 03 85 01
80A8- 4C 16 80 A6 0E EC 04 80
80B0- 30 12 48 A2 00 BD 95 81
80B8- F0 06 20 9D 81 E8 D0 F5
80C0- 20 6B 81 68 C9 22 D0 08
80C8- A5 09 49 80 85 09 A9 22
80D0- A6 09 D0 1B C9 3A D0 13
80D8- A2 00 BD 8D 81 F0 06 20
80E0- 9D 81 E8 D0 F5 20 6B 81
80E8- 4C 94 80 C9 80 10 1A 29
80F0- 7F C9 20 10 0E 48 A9 5B
80F8- 20 9D 81 68 09 40 20 9D
8100- 81 A9 5D 20 9D 81 4C 94
8108- 80 48 C9 81 D0 02 E6 0A
8110- C9 82 D0 02 C6 0A C9 B2
8118- D0 02 E6 09 AA BC 25 81
8120- 84 0B 24 0B 10 05 A9 A0
8128- 20 9D 81 A9 D0 85 0D A9
8130- D0 85 0C 68 AA A0 00 CA
8138- 10 10 B1 0C E6 0C D0 02
8140- E6 0D C9 80 10 F1 30 F2
8148- A0 00 B1 0C C8 AA 20 9D
8150- 81 8A 10 F6 24 0B 50 05
8158- A9 A0 20 9D 81 4C 94 80
8160- A0 00 B1 00 E6 00 D0 02
8168- E6 01 60 A2 0D 86 0E A6
8170- 0A 10 02 A2 00 E0 06 30
8178- 02 A2 06 CA 30 0E AC 03
8180- 80 88 30 F7 A9 A0 20 9D
8188- 81 4C 81 81 60 8D A0 A0
8190- A0 A0 A0 BA 00 8D A0 A0
8198- A0 A0 A0 A0 00 09 80 20
81A0- ED FD E6 0E 60 40 40 40
81A8- 40 40 40 40 40 40 40 00
81B0- 00 40 40 40 40 40 40 00
81B8- 40 40 40 40 40 00 00 40
81C0- 40 40 40 40 40 00 40 40
81C8- 00 00 40 40 40 40 00 40
81D0- 40 40 40 40 00 40 40 00
81D8- 40 40 40 40 40 40 40 40
81E0- 40 40 40 40 40 00 C0 00
81E8- 00 C0 C0 40 C0 00 00 00
81F0- 00 00 C0 C0 00 00 00 00
81F8- 00 00 00 00 00 00 00 00
8200- 00 00 00 00 00 00 00 00
8208- 00 00 00 00 00 00 00 00
```

©

Program Conversion With Sinclair BASIC And TI BASIC

Julie Knott and Dave Prochnow

Program conversion between BASIC dialects is often easier than imagined. This tutorial demonstrates the compatibility of TI BASIC and Sinclair BASIC and includes helpful tables and sample conversion programs.

Program conversion can be an easy and convenient operation. Virtually every home computer uses BASIC, which, because it's easy to learn and to manipulate, is ideal for ready-made language conversion. However, no two BASICs are created equal. For many years the industry's standard was Microsoft BASIC, then different dialects began to emerge. Manufacturers would use the Microsoft format and introduce nuances and subtleties in the structuring, labeling each of these alterations an "improvement" of BASIC. But many were only changes in the protocol—the manner in which a command is expressed. And it becomes relatively easy to convert BASIC dialects if the major differences are in protocol or syntax.

Two versions of BASIC which lend themselves to such a program conversion are Sinclair BASIC and TI BASIC. Sinclair BASIC, used in the Timex/Sinclair-1000, is unique in that all keywords are single-stroke entries. For example, the P key stands for the PRINT command. (The use of a touch-membrane keyboard dictates this procedural necessity.)

Texas Instruments TI-99/4 and 4A use TI BASIC, which is more conventional in that each individual letter has to be typed—PRINT would require five keystrokes.

There are only slight variations between Sinclair BASIC and TI BASIC, but their similarities allow for easy program conversion. By studying which statements and commands are equivalent for both BASICs, and what substitutions are necessary, program conversion can be relatively simple. Also, you can virtually double your software by translating programs published for the other machines.

For the sake of brevity, the following glossary does not contain all of the keywords in Sinclair BASIC and TI BASIC—only those words which are confusing, complicated, or not directly translatable have been listed. For a more complete listing, consult the appropriate user's manual.

Sinclair BASIC

AND — a logical operator, often used in IF-THEN statements

ACS — function that gives the arc cosine of an angle in radians

ASN — function that gives the arc sine of an angle in radians

AT — used in a PRINT statement to give a location at which to PRINT

BREAK — stops program execution, key activated and may not be included as a command in a program

CLEAR — deletes all variables from memory

CLS — clears the screen

CODE — a string function used to obtain the numeric value of a given character

CONT — resumes execution of a program following a report code

COPY — copies the contents of the screen to printer

DELETE — erases keywords and characters while programming

FAST — fast mode, a time-saving mode for increased RUN speed

FUNCTION — function mode

GRAPHICS — graphics mode

INKEY$ —used in IF-THEN statements as a conditional statement, executes exclusive of ENTER

LLIST — lists the contents of a program listing to a printer

LOAD — loads a prerecorded program from cassette tape to the computer's memory

LPRINT — PRINTs to printer

NOT — inverts the truth value of an expression

OR — a logical operator, used in conditional statements

PAUSE — creates a time delay while the program is RUNning

PEEK — gives the value of the byte at a specific address in memory

PI — gives the value of PI

PLOT — draws a pixel at a given location

POKE — puts a numeric value into memory at a specific address, erasing the previous one

SCROLL — scrolls the screen up one line, eliminating the top line

SLOW — slow mode, the standard operating mode

UNPLOT — erases a pixel at a given location

USR — calls a machine language routine at a specific memory address

TI BASIC

APPEND — an open mode, allows data to be added at the end of the existing file

ASC — ASCII value or character code

BASE — option base

BREAK — sets breakpoints in a program, used for error checking

BYE — erases memory, returns to title screen

CALL — special subprogram to obtain color and sound

CLOSE — closes the association between a file and a program

CONTINUE (CON) — continues a program after a breakpoint

DATA — stores data

DEF — defines user-established functions in a program

DELETE — removes a program or data file from a filing system

DISPLAY — prints on screen only

ELSE — conditional part of IF-THEN/ELSE statement

END — terminates program, similar to STOP

EOF — End-Of-File, determines if the end of a file has been reached on an accessory device

FIXED — files with a specified length, used with RELATIVE or SEQUENTIAL

INTERNAL — file type recorded in machine language

NUMBER (NUM) — automatic line number generator

OLD — loads a previously SAVEd program

ON — a conditional numeric expression, used with ON-GOTO or ON-GOSUB

OPEN — prepares to use data files stored in accessory device

OPTION — option base, sets lower limit of array subscripts to 1 instead of 0

OUTPUT — transfers data out of a program

PERMANENT — file life

POS — position

READ — reads data in DATA statements

REC — points to a specific record in a RELATIVE file

RELATIVE — defines a file with FIXED

RESEQUENCE (RES) — reassigns line numbers

RESTORE — identifies which DATA to use with the next READ

SEG$ — string segment, substring

SEQUENTIAL — defines a file, used with FIXED or VARIABLE

SUB — part of GO SUB

TRACE — outlines the order that statements will be performed when the program is RUN

UNBREAK — removes breakpoints

UNTRACE — cancels TRACE

UPDATE — an open mode, for reading and writing into files

VARIABLE — defines a varying length file, used with SEQUENTIAL

Special Subprograms Used With Graphics And Sound In TI BASIC

Each subprogram is preceded by CALL (for example, CALL CLEAR)

CLEAR — erases the entire screen

COLOR — specifies screen character colors

SCREEN — changes screen color

CHAR — defines user-special graphic characters

HCHAR — places a character and repeats it horizontally

VCHAR — similar to HCHAR except repetition is vertical

SOUND — produces tones and noises of different duration, frequency, and volume

GCHAR — reads a character anywhere on the screen

KEY — transfers character directly from keyboard to program without ENTER

JOYST — inputs data with remote controllers

Easy Conversions

Many of the commands and statements of these two BASICs are directly translatable. Table 1 shows the direct BASIC equivalents for Sinclair BASIC and TI BASIC. The only major differences between these two dialects are in their nomenclature.

Several dialects of BASIC have an ON-GOTO statement expressed as:

```
ON x GOTO w,y,z
```

where x is the value of a numerical expression and w, y, and z are line numbers. This statement is available in TI BASIC, but not in Sinclair BASIC. Through the use of conditional expressions, the

Sinclair BASIC substitution is:

GOTO (w AND x=1)+(y AND x=2)+(z AND x=3)

The operators AND and OR would make this possible.

The translation of many program lines requires only the replacement or substitution of a word unique to that particular BASIC. Several of the more common functions and statements are evaluated in this manner in Table 2. The following Sinclair BASIC line will await the pressing of the Y key, exclusive of ENTER:

```
100 IF INKEY$ <> "Y" THEN GOTO 100
```

To perform the same statement in TI BASIC, replace INKEY$ with the KEY subprogram, as follows:

```
100 CALL KEY(0,K,Z)
110 IF K<>89 THEN 100
```

The main difference is in the structuring. The KEY subprogram (subprograms are obtained with CALL) uses three variables to establish where the key is originating, its ASCII code, and its status. In this example the ASCII code of 89 represents the Y character.

TI BASIC has the ability to store expressions and assign values to these variables with the statements DATA, READ, and RESTORE (see the glossary). Vast arrays can be developed and initialized with this method. Sinclair BASIC is not directly convertible with DATA, READ, and RESTORE. A large battery of LET statements *could* crudely handle the data. Alternatively, a properly DIMensioned INPUT statement allows the creation of such an array. Upon completion, the INPUT statements are removed and a GOTO command is used for program starting (RUN erases the variable array).

String Handling

Strings can be equally bothersome. Slicing will supply usable substrings in Sinclair BASIC. A string expression's parameters govern the start and finish of the slice. No special statement is necessary:

A$(x TO z)

with x representing the starting number and z the finish. For example:

"COMPUTE" (4 TO 7) = "PUTE"

The statement SEG$ (A$,x,y) in TI BASIC has the same result, but, again, with different nomenclature. X is the number of the start for the substring and Y is the length of the substring. For example:

A$="COMPUTE"
SEG$ (A$,4,4)="PUTE"

While string slicing is easily translated, the TI BASIC user-defined function is not. DEF allows the definition of functions within a program.

DEF X$="Y"

The string function's name is X and the string expression is Y. VAL and string variables can be user-defined in Sinclair BASIC.

LET X$="Y"
VAL X$= Y

This is a very limited and a "sometimes-maybe" proposition. DEF has the ability to also handle numeric functions. This ability, as well as using parameters in argument evaluation, is beyond VAL's means.

When attempting a program conversion you may run across a few Sinclair BASIC terms that are completely unfamiliar to you. The terms USR, PEEK, and POKE are not procedures for the examination of some strange alien creature. They are primarily associated with direct access to memory. To call a machine language routine that begins with a specific address, USR is used. This will start a machine language program running. POKE is used by the T/S-1000 to store a numeric value at a specific address in the computer's memory. For example:

POKE 17529, 38

POKEs the value 38 into address 17529. Conversely, the PEEK command is used to read certain addresses to see what is stored there. The PEEK command is followed by the address to be PEEKed.

PRINT PEEK 17529

would PRINT the number 38. When you are translating a program from Sinclair BASIC which contains USR, PEEK, and POKE statements, you must find out what they accomplish and then interpret that into TI BASIC.

PRINTing on the screen is accomplished by a blending of line and row markers. Memory conservation techniques notwithstanding, PRINT can be used to move the PRINT line. For example:

PRINT
PRINT
PRINT "COMPUTE"

Sinclair BASIC also allows the movement of PRINT with AT and TAB.

PRINT AT x,y

and

PRINT TAB y

TAB moves the PRINT position a prescribed number of spaces to the right. Even though TAB is present in TI BASIC, the vocabulary is different. Line changes are accomplished with colons (:). Duplicating the above examples,

PRINT::...(x) TAB (y)

and

PRINT TAB (y)

X is the number of colons necessary to equal the value of the line number (x) in the Sinclair BASIC example.

The Timex/Sinclair lacks color and sound features, but these features are of importance on the TI-99/4. TI BASIC's color and sound statements are subprograms that begin with CALL. Clever usage of Sinclair BASIC's character set can duplicate some of these color combinations. As a rule, however, TI BASIC CALL subprograms should be removed and not directly substituted in a program conversion to Sinclair BASIC. This allows concentration on the program's more important graphics. Consultation with Texas Instruments' *User's Reference Guide* will provide the proper protocol for development and inclusion of color and sound subprograms in a Sinclair BASIC converted to TI BASIC program.

To illustrate the principles of program conversion, examine these sample programs. While each program is unique in its results, the approach is similar and convertible. The purpose of this program is to display the entire character set along with the character codes.

T/S-1000 Version

```
10 FORA=0 TO 255
20 LET A$ = CHR$ A
30 PRINT AT 10,13; A
40 PRINT AT 7,10; A$;"{6 SPACES}"
50 PRINT AT 7,17; A$;"{6 SPACES}"
60 PRINT AT 13,10; A$;"{6 SPACES}"
70 PRINT AT 13,17; A$;"{6 SPACES}"
80 NEXT A
```

TI-99/4 Version

```
100 FOR A=32 TO 127
110 B$=CHR$(A)
120 CALL CLEAR
130 CALL SCREEN(2)
140 PRINT TAB(13);B$;TAB(18);B$
150 PRINT
160 PRINT TAB(14);A
170 PRINT
180 PRINT TAB(13);B$;TAB(18);B$
190 PRINT ::::::::
200 FOR S=3 TO 16
210 CALL SCREEN(S)
220 CALL SOUND(400,110+80*(S-3),1)
230 NEXT S
240 NEXT A
```

In line 10 of the Timex/Sinclair example, a loop establishes the number of character codes to be examined (the entire character set is 0 to 255). Note that the characters with codes 67–127 cannot be printed and will show on the screen as question marks. Lines 20 and 30 PRINT the code or numeric value for each character. The arrangement of the printed characters is defined in lines 30–40. In this way, you can easily interpret the delay, and read the code value and the character almost simultaneously. This program will RUN until BREAK is pressed.

Table 1:
Reference Chart Of BASIC Equivalencies

Sinclair BASIC	= TI BASIC
ABS	ABS
ATN	ATN
CHR$	CHR$
CODE	ASC
COS	COS
DIM	DIM
EXP	EXP
FOR	FOR
GOSUB	GOSUB or GO SUB
GOTO	GOTO or GO TO
IF	IF
INPUT	INPUT
INT	INT
LEN	LEN
LET	LET
LN	LOG
LOAD	OLD
NEW	NEW
NEXT	NEXT
PRINT	PRINT
RAND	RANDOMIZE
REM	REM
RETURN	RETURN
RND	RND
RUN	RUN
SAVE	SAVE
SGN	SGN
SIN	SIN
SQR	SQR
STEP	STEP
STOP	STOP
STR$	STR$
TAB	TAB
TAN	TAN
THEN	THEN
TO	TO
VAL	VAL
CLS	CALL CLEAR

Table 2:
Substitution Chart For BASIC Nonequivalents

Sinclair BASIC	= TI BASIC
NEW	BYE
PRINT	DISPLAY
GOTO (W AND X=1) +(Y AND X=2)+(Z AND X=3)	ON X GOTO W,Y,Z
IF X THEN GOTO Y	IF X THEN (Y)
LET X=Y+Z	LET X=Y+Z or X=Y+Z
PAUSE or FOR X=Z TO Y NEXT X	FOR X=Z TO Y NEXT X
INKEY$	CALL KEY
A$(X TO Z)	SEG$(A$,X,Y)
PI	4*ATN (1)
LET X$="Y" VAL X$=Y	DEF X=Y
STOP	END or STOP
PRINT AT X,Y	PRINT:...(X) TAB(Y)
PRINT TAB Y	PRINT TAB(Y)
ASN 1	π /2 or 4*ATN(1)/2 or 2*ATN(1)
IF X=Y THEN GOTO A GOTO Z	IF X=Y THEN A ELSE Z

©

Commodore 64 ROM Generations

Jim Butterfield, Associate Editor

Commodore products are often subject to changes in logic. Not marketing logic or pricing logic (although they change too), but the internal logic that drives the machines: the programs in ROM. This has been true of PET/CBM and various disk systems. This article traces differences in two major ROM releases of the Commodore 64 computer, plus a third released with the SX-64 portable computer.

Two Environments

The first 64s used ROM set 1. Before releasing a European version of the 64, Commodore developed ROM set 2. ROM 2 is unique in that it's the same for North America and Europe, yet recognizes and copes with differences between the two environments. More on that later.

Programs developed on ROM set 1 sometimes didn't seem to work on ROM set 2. This was particularly true when the screen was set up using a POKE statement. For example, a user clearing the screen and then typing the command POKE 1500,1 will print a letter A around the middle of the screen, but with ROM 2 this letter is "invisible." Many games and educational programs using the screen this way couldn't make the transition from ROM 1 to ROM 2; attractive graphics would become invisible and the effect would be lost.

I have met a third ROM recently; it's used in the SX-64 portable computer. There are small differences: For example, disk activities are given preference over tape, and screen POKEs are once again legal.

In all cases, the BASIC language in ROM is not changed (addresses $A000 to $BFFF). All changes are in the Kernal ROM, which resides at addresses $E000 to $FFFF.

All three ROM sets are very similar; the differences are largely cosmetic. Sometimes, of course, cosmetic differences are enough to prevent a particular program from working in a satisfactory manner; but there's a strong bond between all models I have examined.

The Tape Pause

When you give a tape LOAD command, the computer blanks the screen and searches for a program "header" on the tape. When it finds a program, it reports the name with a message, FOUND XXXXX, unblanks the screen, and waits. When you touch a key (preferably the Commodore Logo key), the screen blanks once again and the program starts to load.

ROM 1 waits forever. If you don't press a key, it keeps waiting. ROM 2, however, waits only a few seconds and then proceeds with the program load activity. ROM 3 for the SX-64 doesn't have a cassette tape connection, so it doesn't do either.

Why does the screen need to blank? Here's the reason: The screen interferes very slightly with the processor. Roughly once every 1/2000 second, the processor chip is stopped briefly to allow the video chip to get extra information from memory. This is no hardship except when we need to read or write tape.

When cassette tape is active, the processor needs to time events precisely. It can't afford to miss even the brief time lapse that the video chip might cause. So it turns the screen off in order to get the most efficient timing "edge."

Technical note: The "Find Tape Header" subroutine at $F761 is changed in ROM 2 so that it calls a new subroutine at $E4E0 to allow time-out. The same coding is used in the SX-64 ROM, but it's not useful since this machine can't use tape.

Screen Clear

When ROM 1 clears the screen, it sets the foreground color of all screen locations to white. As a result, it's easy to POKE screen memory and have white characters appear.

ROM 2 changed all that. When the screen clears, the foreground color of all characters is set to the background color. If you POKE to an unused location, you'll end up printing blue on blue, which makes it invisible. The character is indeed there: You can see it if you place the cursor over that position. But it's not much use to the viewer.

Commodore may have done this to reduce screen "sparkle"—colored or white flashes that appear randomly on the screen. Whatever the reasoning, it caused writers of software some anguish if their existing programs POKEd the screen a good deal. Many Commodore demonstration programs lost their appeal on the new machines. All programs would still run, but the screen wouldn't look right.

With the new SX-64 ROM, we're back to allowing screen POKEs. It may be too late for software writers, but when the SX-64 clears the screen, it sets the foreground color of all screen locations to the cursor color. That's better than ROM 1, which sets white only—you have a chance to choose the POKE color.

Technical note: The Clear-a-Line subroutine at $E9FF was changed slightly to call a new subroutine at $E4DA; this sets character color to background color on ROM 2. On SX-64 ROM, character color is set to the value from $0286, the current "cursor" color.

Different Crystal Speeds

ROM 1 was designed for North American use. ROM 2 was designed for worldwide use, and considerable thought was put into creating a universal design. When power is applied to the computer, ROM 2 does some interesting detective work.

Very early in the game, ROM 2 set the raster interrupt to fire at scan line 622. Here's the trick: There is no line 622 on North American sets; if the interrupt signal fires, we must be elsewhere.

Depending on the continent, the ROM sets up timing for the clock and RS-232 transmission. What's happening here is that the two different types of machine are driven at different "crystal" speeds, and the program must compensate for this to allow consistent overall speed.

The programmer on a ROM 2 system must keep in mind that the raster interrupt register in the video chip has already been used by the system; it cannot be assumed to be zero.

Technical note: The table at $ECB9 which sets up the video chip has been changed to include the raster interrupt. The Power-Up Reset program itself has been changed at $FCFB by the insertion of a call to a new subroutine at $FF5B. If line 622 (Europe) is detected, address $02A6 is set to 1 to signal "European System." This new location, $02A6, is used to set up the timer which creates "jiffies"—1/60-second interrupts. It will also be checked if the RS-232 channel is opened, and timing information extracted from the appropriate table.

Small Stuff

ROM 1 had troubles if you tried to PRINT# to a device that wasn't there; ROM 2 has its act together a little better.

SX-64 ROM identifies itself with a new message: SX-64 BASIC V2.0, in case you didn't notice that it was an SX-64 you had.

If you hold down SHIFT and press RUN/STOP on the SX-64, you'll get a load/run from *disk*; the screen reads LOAD":*",8 ... RUN. This data is stored in an area of memory that usually contains the message PRESS PLAY, but you won't be using the cassette this time so you won't miss that message. Any attempt to use a cassette on the SX-64, by the way, will result in an ILLEGAL DEVICE NUMBER message.

The differences are not great. Most users will spot only the tape pause and the screen POKE as operational differences.

Serious programmers will appreciate the fact that changes have been made as "patches," which means that previous entry points have not moved; they are still in the same places that they used to be. A call to a machine language subroutine at a given location will still be good.

There are still things that many users would like to see improved in Commodore 64 BASIC and Kernal. In particular: The INPUT statement is uncomfortable at times, and certain types of screen editing work awkwardly. You may have a wish list of your own. It seems quite likely that we'll see another ROM system one of these days.

Coming Soon

Commodore is said to be working on new ROM systems for the 64 and its peripherals. Compatibility is expected to be retained with previous ROMs, but certain operational annoyances will be eliminated.

Watch for a new Kernal ROM—we expect it to be coded 901227-03. It will fix up a couple of problems associated with screen usage.

The Commodore 64, like the VIC-20, behaves oddly if an INPUT statement is written with a lengthy prompt; if the prompting message is long enough, the user input will need to be typed onto the next line of the screen. In such a case, the computer receives a peculiar input: As well as reading what the user has typed in, it reads its own prompt message.

A more serious problem arises if a user types in a line longer than 80 characters, and then backs up using the Delete key. The too-long line goes beyond two rows on the screen, of course; but when the user backs up, the computer *might* stop working.

The above problems are expected to be fixed when the new version 3 chip is released. In addition, some of the above-noted changes for the SX-64 will also be implemented—for example, screen POKEs.

Commodore is also said to be working on new logic for printers and disk units. Watch for them, too.

Commodore 1541 Generations

Tracking the generations of Commodore's 1541 disk drive is not unlike reading a mystery novel. Unfortunately for 1541 owners, Commodore so far has not written the last page in which the mystery is revealed, so we can only examine the clues and speculate.

Clue No. 1: The original 1541 had a "long" circuit board which extended the length of the drive. This board probably was the same as was in the 1540 drive, predecessor to the 1541.

Clue No. 2: Both the 1540 and the original version of the 1541 had white cases.

Clue No. 3: Later versions of the 1541 have brown cases, and a "short" board which extends about half the length of the drive. Our sources tell us that the short board is a redesigned long board and that when the circuit board was redesigned, timing problems showed up in the drive.

Clue No. 4: ROM chips bearing four different part numbers have been seen in 1541 drives. During a teleconference on the Commodore Information Network on March 29, 1984, a Commodore Research & Development representative gave the part number of the latest ROM as 901229-05. (The suffix 05 indicates the ROM version.) ROM chips with suffixes 01, 02, and 03 also have been seen in 1541 drives.

Clue No. 5: During the teleconference, the Commodore representative said that one of the changes incorporated into the 05 ROM version had to do with the serial bus. (Peripherals such as the 1541 and the 1525 printer connect to the Commodore 64 through the serial bus.)

Clue No. 6: Owners of the 1541 have reported problems when trying to use two 1541s; occasionally, when a program accesses one of the drives, the system locks up. Problems also have been reported involving lockup on systems with one 1541 drive and the Commodore 1526 and MPS-801 printers.

Clue No. 7: 1541 users report an intermittent problem when saving files to disk using the replace option (SAVE "@0:filename",8). Instead of replacing the intended file, the drive's operating system writes over another file on the disk, and changes the directory pointers so that the intended file is no longer accessible. A similar problem has been reported in the Commodore 4040 drives. At the teleconference, the Commodore representative said he'd never experienced this problem. However, he also said that the 4040 and 1541 used the same basic operating system.

Clue No. 8: A technical representative with Integrated Computer Repairs (ICR), of Santa Mesa, California, told us that his company repairs and updates the 1540 and 1541 drives. ICR claims that merely replacing the ROM chip with an 05 version is not enough; they also make hardware changes, modifying the short circuit board.

Clue No. 9: Overheating problems have been reported with the 1541. After the drive has been on for several hours, some users report input-output errors and other problems.

Clue No. 10: In the past, Commodore representatives have said that the 1541 ROM changes were "mainly cosmetic."

Clue No. 11: ICR claims that the drives they have updated no longer have lockup problems. It is not clear whether their update solves the save-with-replace problem.

Clue No. 12: COMPUTE! made several telephone calls to Commodore Business Machines, Inc., asking Commodore to respond to the above items. Ms. Susan West, of the Public Relations Department, promised to find a technical representative who could answer our questions. We never heard from a technical representative, and Ms. West failed to return our subsequent calls, the last placed as this article was going to press.

So, it seems that Commodore has issued at least four different versions of the 1541 disk drive, for reasons which are known only to them. And problems may exist (or have existed) not only in the ROM chips, but also in the board circuitry. Finally, it appears that Commodore is unwilling to help us solve the mystery. ©

Atari MacroDOS: Part 2

Jerry Allen

Last month we introduced "Atari MacroDOS" and presented a BASIC program which loads the MacroDOS machine language. This month we'll look at some technical details of MacroDOS and present a disassembly of the program.

Assembler users can alter the MacroDOS commands table (CMDTAB) if they so desire. Just remember to change lines which check for command input to reflect the new command letter. Also, revise TAB1 if necessary.

You can append another AUTORUN.SYS program to the end of MacroDOS, such as a menu loader for BASIC.

Assuming you have MacroDOS up and running as AUTORUN.SYS, enter DOS, then load the other AUTORUN.SYS from another disk. SAVE with APPEND ("D:AUTORUN.SYS"). Enter the beginning and ending addresses at the @ prompts. If necessary, return to the cartridge and POKE or otherwise change the INIT and RUN addresses. Return to DOS and SAVE with append again to pick up the addresses just altered.

Finding Load Addresses

If you can't figure out the load addresses, use this program:

```
1Ø OPEN#2,4,Ø,"D:YOURPROG.OBJ
2Ø FOR I=1 TO 6
3Ø GET#2,A
4Ø PRINT A
5Ø NEXT A
6Ø CLOSE#2
```

The first two bytes should be a header of 255 ($FF). The next four bytes will be the beginning and ending addresses of the load (two-byte numbers in low byte, high byte format).

If the file loads to multiple address areas (including RUN and INIT) after the first block of memory is loaded, OS checks for a new header of 255,255. If it is there, the header is ignored, and the next four bytes will be the new from-to load addresses.

Loading With Page 6

Loading RUN with page 6 (1536) would look like:

$E0(224),$02(2),$E0(224),$02(2),$00(0),$06(6)

(without the header). With a little math and modification of the program, you could find all the load addresses of any compound load file.

If you don't want to type the programs in, send $3, and a disk or tape with an SASE mailer.

Jerry Allen
1906 Carnegie #E
Redondo Beach, CA 90278

MacroDOS, Machine Language Source Code

Refer to the "Automatic Proofreader" article before typing this program in.

```
Ø2ØØ ;EQUATES
Ø21Ø ICBC   = $342
Ø22Ø ICBAL  = $344
Ø23Ø ICBAH  = $345
Ø24Ø ICBLL  = $348
Ø25Ø ICBLH  = $349
Ø26Ø ICBAX  = $34A+16
Ø27Ø MEMLO  = $2E7
Ø28Ø LBUF  = $58Ø
Ø29Ø INBUF  = $F3
Ø3ØØ CIX  = $F2
Ø31Ø FRØ  = $D4
Ø32Ø FR1  = $EØ
Ø33Ø GETR  = $Ø5
Ø34Ø GETC  = $Ø7
Ø35Ø PUTC  = $ØB
Ø36Ø PUTR  = $Ø9
Ø37Ø OPEN  = $Ø3
Ø38Ø CLOSE  = $ØC
Ø39Ø AXIO  = $ØC
```

```
0400 AXOUT = $08
0410 AXAP = $09
0420 AXDR = $06
0430 FR0Z = $DA44
0440 IFP = $D9AA
0450 FPI = $D9D2
0460 FPASC = $D8E6
0470 FMOVE = $DDB6
0480 ASCFP = $D800
0490 LO = $00FF
0500 HI = $0100
0510 CIO = $E456
0520 WARMST = $E474
0530 DOSVEC = $0A
0540 DOSINI = $0C
0550 OLDDOS = $179F
0560 JMPINI = $1705
0570 JMPRUN = $1708
0580 ;
0590 ;
0600   *= $1CFC ;change this addr
 for
0610 ;larger versions of DOS2.0S
0620 ;
0630 ;
0640 ST JSR CLSE ;to be sure
0650   STY CIX ;set f.p. pointer
0660   DEY
0670   STY $2E3;clear INIT/RUN
0680   STY $2E1
0690   STY $FF ;reset load flag
0700   LDX #LBUF&LO ;init flt. pt
 .
0710   LDY #LBUF/HI
0720   STX INBUF
0730   STY INBUF+1
0740 DISCMD LDY #TAB1-CMDTAB-1
0750 L1 TYA ;DISPLAY COMMANDS
0760   PHA
0770   LDA CMDTAB,Y
0780   JSR PRINT
0790   PLA
0800   TAY
0810   DEY
0820   BPL L1
0830   LDA #AXOUT ;init aux
0840   STA ICBAX
0850   JSR GTREC ;get command
0860   LDY #4 ;GET COMMAND
0870   LDA LBUF
0880 L2 CMP TAB1,Y
0890   BEQ SPECMD
0900   DEY
0910   BPL L2
0920 DIR CMP #'D ;DIRECTORY
0930   BNE WDS
0940   LDA #06
0950   STA ICBAX
0960   JSR ASKDN ;drive #?
0970   JSR OPN    ;open
0980 L3 LDX #$10
0990   JSR GTREC ;get formatted l
 ine
1000   JSR PTREC0 ;print it
1010   BPL L3
1020 SPECMD LDA TAB2,Y ;SPECIAL C
 MDS
1030   PHA ;save cmd
1040   CMP #$FE ;check if format
1050   BEQ FMT
1060   JSR PFN
1070 EX PLA ;retrieve command
1080   JSR EXCMD ;do it
1090   BPL ST
1100 FMT LDA #'? ;FORMAT
1110   JSR PRINT ;sure?
1120   JSR GTREC
1130   LDA LBUF
1140   CMP #'Y ;is there a yes?
1150 B1 BNE ST ;if not start over
1160   JSR ASKDN ;get drive#
1170   BMI EX ;execute
1180 ASKDN LDA #'D ;GET DRIVE#
1190   JSR PRINT
1200   LDA #'#
1210   JSR PRINT
1220   JSR GTREC
1230   LDA LBUF
1240   STA ADRDIR+1 ;change D#
1250   STA ADRDOS+1 ;just in case
 WDS
1260   LDY #6
1270 L4 LDA ADRDIR,Y ;move filena
 me
1280   STA (INBUF),Y
1290   DEY
1300   BPL L4
1310   RTS
1320 PFN LDA #'F ;PROMPT FILENAME
1330   JSR PRINT
1340   LDA #'N
1350   JSR PRINT
1360   LDA #'?
1370   JSR PRINT
1380   JMP GTREC
1390 WDS CMP #'W ;WRITE DOS.SYS
1400   BNE LOD
1410   JSR ASKDN
1420   INY
1430 L5 LDA ADRDOS,Y
1440   STA (INBUF),Y
1450   DEY
1460   BNE L5
1470   JSR OPN
1480   BPL B1
1490 LOD CMP #'L
1500   BNE SAV
1510   STA $FF
1520   JSR PFN
1530   LSR ICBAX ;8>4
1540   JSR OPN
1550 L6 JSR GETCR2 ;get hdr in pa
 irs
1560   LDA #$FF ;check headr and-
1570   CMP FR0 ;disregard $FF'S
1580   BNE SK2
1590   CMP FR0+1
1600   BEQ L6
1610 SK2 JSR FMOVE ;FP0 to FP1
1620   JSR GETCR2
1630   JSR SUBTR ;subtr HI-LO &ex
 ecute
1640   JSR CHKIN
1650   BEQ L6
1660 CHKIN LDA $2E3
1670   BEQ SK12
1680   JSR JMPINI
1690   LDA #0
1700   STA $2E3
1710 SK12 RTS
1720 SAV CMP #'S ;SAVE FUNCTION
1730   BNE RUN
1740   JSR PFN
1750   LDY #$FF ;check if append
 (/)
1760 L7 INY
1770   LDA (INBUF),Y
1780   CMP #'/
```

```
1790   BNE SK4
1800   INC ICBAX ;9=append
1810   LDA #$9B
1820   STA (INBUF),Y
1830 SK4 CMP #$9B
1840   BNE L7
1850   JSR OPN ;open for write
1860   LDA #$FF ;start headr
1870   STA FRØ
1880   STA FRØ+1
1890   JSR PUTCR2 ;write it
1900   JSR INPCON ;get from#
1910   JSR PUTCR2 ;write it
1920   JSR FMOVE  ;store it
1930   JSR INPCON ;get to#
1940   JSR PUTCR2 ;write it
1950   JSR SUBTR  ;find len and s
  ave
1960   BPL B2 ;the rest
1970 RUN CMP #'@ ;RUN
1980   BNE CART
1990   JSR INPCON ;get #
2000   JMP (FRØ)  ;jump indirectl
  y
2010 CART CMP #'C ;CARTRIDGE
2020   BNE ADOS
2030   JMP WARMST
2040 ADOS CMP #'! ;ESC TO ATARI D
  UP
2050   BNE HEX
2060   LDA #$40 ;reset DOSINI for
   no-
2070   STA DOSINI ;trouble later
2080   LDA #$15
2090   STA DOSINI+1
2100   JSR $1540 ;fast reset DOSV
  EC
2110   JMP OLDDOS
2120 HEX CMP #'$ ;HEX TO DEC
2130   BNE DEC
2140   JSR HASCI ;hex to int
2150   JSR IASC ;int to dec
2160   BPL B2
2170 DEC CMP #'. ;DEC TO HEX
2180   BNE B2
2190   JSR DASCI ;dec to int
2200   JSR IHASC ;int to hex
2210 B2 BPL LSTCNC+2
2220 OPN LDA #OPEN ;IOCB MAIN SET
  UPS
2230 EXCMD LDX #$10
2240   BNE GTR2
2250 PTRECØ LDX #Ø
2260 PTREC LDA #PUTR
2270   BNE GTR2
2280 GETRECØ
2290 GTREC LDA #GETR
2300 GTR2 STA ICBC,X
2310   LDA #$1E ;max rec length
2320   STA ICBLL,X
2330   LDA #LBUF/HI
2340   STA ICBAH,X
2350   LDA #LBUF&LO
2360 PGIN STA ICBAL,X
2370   LDA #Ø
2380   STA ICBLH,X
2390 ICB JSR CIO ;let the OS take
  over
2400   BMI ERR
2410 RTN2 RTS
2420 PUTCR2 LDA #PUTC
2430   BNE GETC2
2440 GETCR2 LDA #GETC
2450 GETC2 LDX #$10
2460   STA ICBC,X
2470   LDA #2
2480 JG STA ICBLL,X
2490   LDA #Ø
2500   STA ICBAH,X
2510   LDA #FRØ
2520   BNE PGIN
2530 CLSE LDA #CLOSE
2540   BPL EXCMD
2550 PRINT STA FRØ
2560   LDX #Ø
2570   LDA #PUTC
2580   STA ICBC,X
2590   LDA #1
2600   BPL JG
2610 ERR CPY #$03 ;ERR next read
OK
2620   BEQ RTN2
2630   CPY #$88 ;ERR EOF OK too
2640   BEQ CINI
2650   TYA      ;store ERR
2660   PHA
2670   LDA #$C5 ;inverted E for e
rror
2680   JSR PRINT
2690   JSR FRØZ ;clear FPØ
2700   PLA      ;retrieve ERR
2710   STA FRØ
2720   JSR IASC ;int to dec
2730 LSTCNC PLA ;clr stack of ret
urn
2740   PLA
2750   JMP ST ;do not pass GO
2760 CINI LDA $FF
2770   BEQ LSTCNC
2780   JSR CHKIN
2790 JRUN LDA $2E1
2800   BEQ LSTCNC
2810   JSR JMPRUN
2820   BNE LSTCNC
2830 INPCON LDA #'@ ;HEX OR DEC#
TYPE
2840   JSR PRINT ;the @ means AT/
TO
2850    JSR GTREC
2860    LDY LBUF
2870    CPY #'.
2880    BEQ DASCI
2890    CPY #'$
2900    BEQ HASCI
2910    BNE ERR ;bad input
2920 DASCI JSR ASCFP ;DEC TO INT
2930    JMP FPI
2940 HASCI JSR FRØZ ;HEX TO INT
2950   LDY #1
2960 LB LDA (INBUF),Y
2970   CMP #$9B
2980   BEQ RTN
2990   SEC ;convert each digit
3000   SBC #$30
3010   CMP #$ØA
3020   BMI SK7
3030   SBC #7
3040 SK7 LDX #4 ;times 16
3050 LA ASL FRØ
3060   ROL FRØ+1
3070   DEX
3080   BNE LA
3090   ORA FRØ ;add in new bits
3100   STA FRØ
3110   INY
3120   BPL LB
3130 RTN RTS
```

```
3140 IHASC LDY #0 ;INT TO HEX SUB
     RT
3150     LDX #1
3160 LC LDA #$F0 ;hi mask
3170     AND FR0,X
3180     LSR A    ;roll into low b
     its
3190     LSR A
3200     LSR A
3210     LSR A
3220     JSR CONVH ;go conv to dig
     it
3230     LDA #$0F ;lo mask
3240     AND FR0,X
3250     JSR CONVH
3260     DEX
3270     BPL LC ;one more time
3280     BMI LE ;set eol and rtn
3290 CONVH CMP #$0A ;INT TO HEX D
     IGIT
3300     BMI SK9
3310     ADC #6 ;carry set
3320 SK9 ADC #$30 ;carry clr
3330     STA (INBUF),Y
3340     INY
3350     RTS
3360 IASC JSR IFP ;INT TO DEC
3370     JSR FPASC
3380 AREC LDY #0 ;CLR HI BIT/MAKE
     REC
3390 L9 LDA (INBUF),Y
3400    INY
3410    CMP #$80 ;find hi bit char
3420    BMI L9
3430    AND #$7F ;mask it
3440    DEY
3450    STA (INBUF),Y
3460    INY
3470 LE LDA #$9B ;set EOL
3480    STA (INBUF),Y
3490    JMP PTREC0
3500 SUBTR INC FR0 ;inclusive so
     inc
3510     BNE SK5 ;TO address
3520     INC FR0+1
3530 SK5 LDX #$10
3540     LDA FR1
3550     STA ICBAL,X
3560     LDA FR1+1
3570     STA ICBAH,X
3580     SEC ;CALC LENGTH
3590     LDA FR0
3600     SBC FR1
3610     STA ICBLL,X ;STORE AT IOC
     B
3620     LDA FR0+1
3630     SBC FR1+1
3640     JMP ICB-3 ;exec same cmd
     as last
3650 ;
3660 ;
3670 INIT JSR $1540 ;SRESET INIT
3680 JI LDA #END&LO
3690    STA MEMLO
3700    LDA #END/HI
3710    STA MEMLO+1
3720    LDA #ST&LO
3730    STA DOSVEC
3740    LDA #ST/HI
3750    STA DOSVEC+1
3760    RTS
3770 ;
3780 ;
3790 CMDTAB ;all spaces are one s
     pace
3800     .BYTE ">",$9B,$9B,"SD",$A
     1
3810     .BYTE " $>",$AE," .>",$A4
3820     .BYTE " SD",$D7," NR",$C0
3830     .BYTE " DO",$CC," VA",$D3
     ,$9B
3840     .BYTE "TR",$C3," TM",$C6
3850     .BYTE " SR",$C5," MN",$D2
3860     .BYTE " *N",$D5," KL",$AA
3870     .BYTE " RI",$C4,$9B
3880 TAB1
3890     .BYTE "RE*UF"
3900 TAB2
3910     .BYTE $20,$21,$23,$24,$FE
3920 ADRDIR
3930     .BYTE "D1:*.*",$9B
3940 ADRDOS
3950     .BYTE "D1:DOS.SYS",$9B
3960 END ;end after boot init exe
     cuted
3970 ;
3980 ;
3990 INIT1 LDA #INIT&LO ;BOOT INI
     T
4000    STA DOSINI
4010    LDA #INIT/HI
4020    STA DOSINI+1
4030    JMP JI
4040 ;
4050 ;
4060    *= $2E2 ;LOAD AND GO INIT
     ADR
4070    .WORD INIT1
4080    .END
```

©

Commodore Garbage Collection Part 2

Jim Butterfield, Associate Editor

Last month, we looked into some of the causes of garbage collection delays, and investigated some of its working mechanisms. It's time to put our knowledge to work by developing some rules.

The following program will help us see the rules by means of examples:

```
100 DIM A$(800)
110 FOR J=1 TO 800
120 A$(J)="A"
130 NEXT J
140 PRINT "X"
150 PRINT FRE(0)
160 PRINT "Y"
```

Rules of Garbage Collection

Rule 1: There are *static* (in place) strings and *dynamic* (created) strings. Only dynamic strings have garbage collection consequences.

Proof: RUN the above program which contains only static strings. There will be no significant delay between the printing of X and Y. Now change line 120 to read:

120 A$(J)=CHR$(65)

RUN once again; there will be a significant pause between the printing of X and Y.

Rule 2: Garbage collection time depends on the number of dynamic strings you *keep*, not what you throw away.

Proof: Change line 120 to read:

120 A$(J)=CHR$(65):A$(J)="A"

RUN the program. Even though we're throwing away a large amount of garbage (the first A$(J)=), there's no significant delay.

Rule 3: Performing a garbage collection saves you no time on the next one.

Proof: Enter line 120 as:

120 A$(J)=CHR$(65)

RUN and note the delay. Now type: GOTO 140. Note that the delay is exactly the same as before; the previous collection saved us no time.

Rule 4: Doubling the number of strings will multiply the delay by 4. Mathematically, we can say that the time varies as the square of the number of strings.

Proof: Change the value of 800 in lines 100 and 110 to 400. RUN and note that the delay between the printing of X and Y drops to one-quarter of the previous time.

This last rule is the killer. You might work out a test program using ten strings, and when your program works satisfactorily expand to one thousand items. But your garbage collection time doesn't increase by a factor of 100; it jumps to 10,000 times the original delay. This could become crippling.

Fixing The Problem

If you know what to look for, you can usually avoid massive garbage collection delays. There's no single technique that will do the job. It's best to investigate what's causing the garbage and decide on the appropriate action to eliminate the problem.

Here's a list of techniques to get around the

garbage collection hang-up.

1. Don't Move Strings Around

Suppose we are writing a program to input several names and sort them into alphabetical order. It would seem logical to move the names so as to put them into the right place. Don't. Use an index array, which contains only numbers: Move the index values, not the strings.

A simple example:

```
100 PRINT "INPUT TEN NAMES"
110 DIM N$(10),I%(10)
120 FOR J=1 TO 10
130 PRINT "NAME";J;
140 INPUT N$(J)
150 I%(J)=J
160 NEXT J
170 PRINT "SORTING..."
180 FOR J=9 TO 1 STEP -1
190 FOR K=1 TO J
200 IF N$(I%(K)) <= N$(I%(K+1)) GOTO 220
210 I=I%(K):I%(K)=I%(K+1):I%(K+1)=I
220 NEXT K,J
230 FOR J=1 TO 10
240 PRINT N$(I%(J))
250 NEXT J
```

The above program uses a bubble sort technique, which is notoriously inefficient; but the point here is that the strings N$(..) are never moved. Thus, there can be no garbage collection. Note that the index array must be initialized before use—see line 150.

2. Clean Up Between Blocks

Suppose you're reading in a large file of students from various classes. For a number of reasons—especially processing convenience and shortage of memory—you don't read in all the students. Instead, you read and process a class at a time.

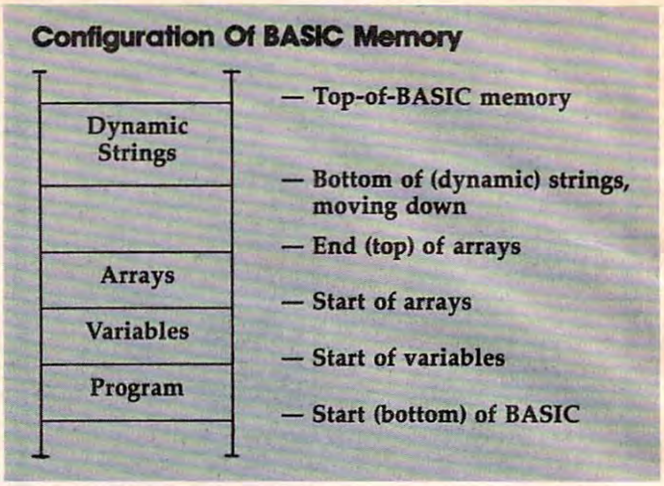
Before reading in the next class, set all student names, to null strings. Now, force a garbage collection with a statement such as Z=FRE(0). There will be few or no strings to keep, so garbage collection will be fast. When the next block of data—the next class—comes in, it will have freshly cleaned memory to use.

3. Do Local Cleanups

Many programs like to build strings from GET statements. The code often looks like this:

```
500 PRINT "TYPE IN YOUR NAME"
530 N=""
540 GET K$:IF K$="" GOTO 540
550 IF K$=CHR$(13) GOTO 600
560 N$=N$+K$
570 GOTO 540
600 REM CONTINUE .....
```

This sort of thing creates a lot of garbage. Every time line 550 is executed, a new N$ is created and the old one is thrown away; and N$ gets bigger and bigger all the time. There's also gar-



bage from K$, but it's only a single character at a time.

If N$ and K$ were our only strings, we'd have no problem. Garbage collection time depends only on what you keep, not what you throw away; and keeping two strings isn't much work. However, if this were part of a program which also had a thousand names and addresses, we'd be in trouble; everything would need to be reclaimed, and the delays would become impractically long.

Local Collection

If we're careful, we can get around this problem by setting the stage for a "local" collection. We might reason as follows: During the above code, N$ and K$ are our only working strings. If we make all the other strings disappear momentarily, we may generate all the garbage we like, since garbage collections will be virtually instantaneous. When we're finished, we must carefully force one last collection to get rid of any leftover garbage, and then make these missing strings reappear.

We can do this by *temporarily* moving the top-of-BASIC pointer down to match the dynamic string pointer. This will fool the garbage collection routine into thinking that there are no dynamic strings except the ones we have just created. But we must remember to put the top-of-BASIC pointer back when the job is finished, or we'll suffer permanent loss of memory.

The top-of-BASIC pointer may be found on the VIC and 64 at addresses 55 and 56. We must save the values there so that we can replace them later, and then use the contents of the string pointer (51 and 52) to change the top-of-BASIC pointer. (In the PET/CBM, the top-of-BASIC pointer is at 52 and 53, and the string pointer is at 48 and 49. We'll show the programming for the VIC/64 below, but you may adjust it for your machine.)

Here's how we would change the above coding to eliminate garbage collection dangers:

```
500 PRINT"TYPE IN YOUR NAME"
510 A1=PEEK(55):A2=PEEK(56)
520 POKE 55,PEEK(51):POKE 56,PEEK(52)
530 N=""
540 GET K$:IF K$="" GOTO 540
550 IF K$=CHR$(13) GOTO 580
560 N$=N$+K$
570 GOTO 540
580 Z=FRE(0)
590 POKE 55,A1:POKE 56,A2
600 REM CONTINUE.....
```

It seems complex, and you must indeed program with great care. But it solves the problem.

4. Use Numeric Values

Who says that everything that seems alphabetic must be a string? A month can be coded 1 to 12; a grade of A to F can be a numeric from 1 to 6.

Where the number of possible strings is limited—a class, a region, an airline—using a numeric system is quite feasible. You can always look up the string you want by using the number as an index and getting the name out of an array.

I wouldn't recommend that we all lose our names and become numbers within the computer. But a little sensible data reduction can save a lot of garbage collection.

5. Brute Force

Sometimes conventional methods fail. Your data consists of a large number of names which have been read in from a file. You need to make changes to a substantial number of these names. There seems to be no way you can control the amount of garbage. What then?

Use The Disk

When all else fails, write out all your strings to disk. Set the strings to null values and force a garbage collection—this will take place instantaneously. Now read them back in to the newly cleaned-up memory.

You can watch the string pointer (addresses 51 and 52 on the VIC/64), and when it seems to be getting near the danger point, initiate this whole operation. At least it will be under your control; you can print a message to the user (TAKE A BREAK WHILE I UNSCRAMBLE MY BRAINS), and may even get the bonus of having generated a data backup or checkpoint in case of loss of power.

And it's a lot better than having the machine go dead for twenty minutes or more.

NEWS&PRODUCTS

Stress Reduction Software

Relax, a computer-controlled biofeedback system aimed at stress reduction, is available from Synapse Software for the Atari, Commodore, Apple, and IBM personal computers.

The system allows the user to observe and measure his or her stress levels on a video monitor or television set, and provides a method to attempt to reduce those levels.

A headband with three electromyograph (EMG) sensors measures tension in the forehead's frontalis muscle. The software converts these measurements into visual patterns designed to monitor the stress level. An audio tape has a program of therapeutic relaxation exercises, and a workbook provides guidelines for reducing stress and establishing a personal stress management profile.

Relax is available for $139.95.

Synapse Software
5221 Central Avenue
Richmond, CA 94804
(415) 527-7751

Apple Educational Games

Methods and Solutions, Inc., has announced its Mindplay line of educational software games that teach children from four years of age and up skills in measurement, following directions, memory, map reading, tactics, vocabulary, grammar, art, and mathematics.

The six educational games in the series include *Bake & Taste,* programs that teach youngsters to measure and follow directions; *Dyno-Quest,* a game of memory, map reading, tactics, and the discovery of dinosaurs; *Picture Perfect,* a joystick-based game that teaches children to draw and to color shapes, designs, and animals; *Race the Clock,* a matching game of words and hidden pictures; *Cat 'n Mouse,* a maze game using word and picture associations; and *Math Magic,* a monster-filled arcade game that teaches addition and subtraction.

The games are priced from $34.95, and are available for the Apple II family of computers and for the IBM PC and PCjr. They will be available for the Commodore 64 this fall.

Methods and Solutions, Inc.
300 Unicorn Park
Woburn, MA 01801
(617) 933-3298

Coleco Adam Data Packs

Victory Software has introduced blank data packs for the Coleco Adam computer. The blank, preformatted tapes store about 250 pages of information.

The tapes are available for a suggested price of $3.98.

Victory Software has also announced its new line of games for the Adam, including *Bounty Hunter* ($19.95), an Old West text adventure game.

Victory Software Corporation
1410 Russell Road
Paoli, PA 19301
(800) 243-1515
(215) 296-3787

Apple, IBM Classroom Software

Classmate, a classroom grading and attendance software package, has been released by Davidson & Associates for the Apple II, IIe and II+, and the IBM PC, PCjr and XT.

The program allows users to enter, modify, and store an unlimited number of class lists for up to 51 students. It stores grades, attendance records and teacher comments, and computes weighted averages, graphs grade distribution, class rankings and final grades, and displays or prints out all records.

The program also can sort by student name or class designation, and can display or print out individual scores, either on a particular assignment or for all assignments.

The program will give out either a single student's or a full class's complete or partial record.

In addition, the program can generate individualized parent and student reports.

Classmate is available for $49.95.

Davidson & Associates
6069 Groveoak Place, #12
Rancho Palos Verdes, CA 90274
(213) 383-9473

Foreign Language Tutorials

Soflight Software, a division of M. P. Computer Services Corporation, has introduced a new product line of foreign language development software.

The programs were designed for the Apple II and IIe, with software for the Atari and IBM PC and PCjr to be available in the future.

One disk drive is required to run the program.

All programs teach 1000 of the most common words in the target language. Where words have more than one meaning, the program allows for those other meanings, along with English translation.

The package retails for $56.95. Languages currently available include Spanish, French, German, Italian, Biblical Hebrew, modern Hebrew, and Arabic. Latin, Russian, Polish, Swedish, and classical Greek will be available in the near future.

Each language program is menu-driven with sequential review, random review, and quiz options.

Soflight Software
2223 Encinal Station
Sunnyvale, CA 94087
(408) 735-0871

Personal Finances Software

A software product designed to help consumers make personal financial decisions has been announced by Electronic Arts. Called *Financial Cookbook*, the program contains "recipes," or formulas, that produce answers about money matters.

Through the program's 32 different recipes, users can figure such data as returns on investments, effective tax shelters and IRAs, effects of inflation, mortgage calculations, and tax rates.

Each recipe asks the user to enter variables, such as interest or inflation rates, and then makes calculations based on those numbers.

Calculations for 11 basic tax shelters available to most consumers are found in the recipes. The instruction manual includes a tutorial, recipe instructions, and index.

Financial Cookbook is available for the entire Apple II line, the IBM PC and PCjr, Commodore 64, and Atari 800.

Suggested retail price is $50.

Electronic Arts
2755 Campus Drive
San Mateo, CA 94403
(415) 571-7171

Text Adventure For Youngsters

Infocom has announced *Seastalker*, an interactive text adventure game for ages 9 and up.

In it, players aboard the specially equipped submarine Scimitar must save the Aquadome, earth's first undersea research station.

Unfortunately, the Scimitar hasn't been tested in deep water, and the crew of the Aquadome may have a traitor in its ranks. If the right course isn't charted, players might end up as shark bait.

Solving hints are included in the game package.

Seastalker is available for the Apple II, Atari, Commodore 64, IBM PC and PCjr, and TI-99/4A at a cost of $39.95.

Infocom, Inc.
55 Wheeler St.
Cambridge, MA 02138
(617) 492-1031

New Product releases are selected from submissions for reasons of timeliness, available space, and general interest to our readers. We regret that we are unable to select all new product submissions for publication. Readers should be aware that we present here some edited version of material submitted by vendors and are unable to vouch for its accuracy at time of publication.

COMPUTE! *welcomes notices of upcoming events and requests that the sponsors send a short description, their name and phone number, and an address to which interested readers may write for further information. Please send notices at least three months before the date of the event, to: Calendar, P.O. Box 5406, Greensboro, NC 27403.* ©

The Automatic Proofreader
For VIC, 64, And Atari

Charles Brannon, Program Editor

At last there's a way for your computer to help you check your typing. "The Automatic Proofreader" will make entering programs faster, easier, and more accurate.

The strong point of computers is that they excel at tedious, exacting tasks. So why not get your computer to check your typing for you?

With "The Automatic Proofreader" nestled in your VIC-20, Commodore 64, or Atari computer, every line you type in will be verified. It displays a special code, called a *checksum*, at the top of the screen. The checksum, either a number (VIC/64) or a pair of letters (Atari), corresponds to the line you've just typed. It represents every character in the line summed together. A matching code in the program listing lets you compare it to the checksum which the Proofreader displays. A glance is all it takes to confirm that you've typed the line correctly.

Entering The Automatic Proofreader

Commodore (VIC/64) owners should type in Program 1. Program 2 is for Atari users. Since the Proofreader is a machine language program, be especially diligent. Watch out for typing extra commas, or a letter O for a zero, and check every number carefully. If you make a mistake when typing in the DATA statements, you'll get the message "Error in DATA statements" when you RUN the program. Check your typing and try again.

When you've typed in The Automatic Proofreader, SAVE it to tape or disk at least twice *before running it for the first time*. If you mistype the Proofreader, it may cause a system crash when you first run it. By SAVEing a copy beforehand, you can reLOAD it and hunt for your error. Also, you'll want a backup copy of the Proofreader because you'll use it again and again—every time you enter a program from COMPUTE!.

When you RUN the Proofreader, the program will be POKEd safely into memory, then it will activate itself. If you ever need to reactivate it (RUN/STOP—RESTORE or SYSTEM RESET will disable it), just enter the command SYS 886 (VIC/64) or PRINT USR(1536) for the Atari.

Using The Proofreader

Now, let's see how it works. LIST the Proofreader program, move the cursor up to one of the lines, and press RETURN. If you've entered the Proofreader correctly, a checksum will appear in the top-left corner of your screen.

Try making a change in the line and hit RETURN. Notice that the checksum has changed. All VIC and 64 listings in COMPUTE! now have a number appended to the end of each line, for example, :rem 123. *Don't enter this statement*. It is just for your information. The rem is used to make the number harmless if someone does type it in. It will, however, use up memory if you enter it, and it will cause the checksum displayed at the top of the screen to be different, even if you entered the rest of the line correctly.

The Atari checksum is found immediately to the left of each line number. This makes it impossible to type in the checksum accidentally, since a program line must start with a number.

Just type in each line (without the printed checksum), and check the checksum displayed at the top of the screen against the checksum in the listing. If they match, go on to the next line. If they don't, there's a mistake. You can correct the line immediately, instead of waiting to find the error when you RUN the program.

The Proofreader is not picky with spaces. It will not notice extra spaces or missing ones. This is for your convenience, since spacing is generally not important. Occasionally proper spacing is important, but the article describing the program will warn you to be careful in these cases.

Nobody's Perfect

Although the Proofreader is an important aid, there are a few things to watch out for. If you enter a line by using abbreviations for commands, the checksum will not match up. This is because the Proofreader is very literal: It looks at the individual letters in a line, not at tokens such as PRINT. There is a way to make the Proofreader check such a line. After entering the line, LIST it. This makes the computer spell out the abbreviations. Then move the cursor up to the line and press RETURN. It should now match the checksum. You can check whole groups of lines this way. Atari users should beware of using ? as an abbreviation for PRINT—they're not the same thing in the Proofreader's eyes.

The checksum is a sum of the ASCII values of the characters in a line. VIC and 64 owners may wonder why the numbers are so small, never exceeding 255. This is because the addition is done only in eight bits. A result over 255 will roll over past zero, like an odometer past 99999. On the Atari, the number is turned into two letters, both for increased convenience and to make the Proofreader shorter. For the curious, the letters correspond to the values of the left and right nybbles added to 33 (to offset them into the alphabet). This number is then stored directly into screen memory.

Due to the nature of a checksum, the Proofreader will not catch all errors. Since $1+3+5=3+1+5$, the Proofreader cannot catch errors of transposition. In fact, you could type in the line in any order, and the Proofreader wouldn't notice. Anytime the Proofreader seems to act strange, keep this in mind. Since the ASCII values of the number 18 (49+56) and 63 (54+51) both equal 105, these numbers are equal according to the Proofreader. There really is no simple way to catch these kinds of errors. Fortunately, the Proofreader will catch the majority of the typing mistakes most people make.

If you want the Proofreader out of your way, just press SYSTEM RESET or RUN/STOP—RESTORE. If you need it again, enter SYS 828 (VIC/64) or PRINT USR(1536) (Atari). You must disable the Proofreader before doing any tape operations on the VIC or 64.

Hidden Perils

The Proofreader's home in the VIC and 64 is not a very safe haven. Since the cassette buffer is wiped out during tape operations, you need to disable the Proofreader with RUN/STOP—RESTORE before you SAVE your program. This applies only to tape use. Disk users or Atari owners have nothing to worry about.

Not so for VIC and 64 owners with tape drives. What if you type in a program in several sittings? The next day, you come to your computer, LOAD and RUN the Proofreader, then try to LOAD the partially completed program so you can add to it. But since the Proofreader is trying to hide in the cassette buffer, it is wiped out!

What you need is a way to LOAD the Proofreader after you've LOADed the partial program. The problem is, a tape load to the buffer destroys what it's supposed to load.

After you've typed in and RUN the Proofreader, enter the following lines in direct mode (without line numbers) *exactly* as shown:

```
A$="PROOFREADER.T": B$="{10 SPACES}": FOR
   X = 1 TO 4: A$=A$+B$: NEXTX
FOR X = 886 TO 1018: A$=A$+CHR$(PEEK(X)):
   NEXTX
OPEN 1,1,1,A$:CLOSE1
```

After you enter the last line, you will be asked to press record and play on your cassette recorder. Put this program at the beginning of a new tape. This gives you a new way to load the Proofreader. Anytime you want to bring the Proofreader into memory without disturbing anything else, put the cassette in the tape drive, rewind, and enter:

OPEN1:CLOSE1

You can now start the Proofreader by typing SYS 886. To test this, PRINT PEEK(886) should return the number 173. If it does not, repeat the steps above, making sure that A$ ("PROOFREADER.T") contains 13 characters and that B$ contains 10 spaces.

You can now reload the Proofreader into memory whenever LOAD or SAVE destroys it, restoring your personal typing helper.

Incidentally, you can protect the cassette buffer on the Commodore 64 with POKE 178,165. This POKE should work on the VIC, but it has caused numerous problems, probably due to a bug in the VIC operating system. With this POKE, the 64 will not wipe out the cassette buffer during tape LOADs and SAVEs.

Program 1: VIC/64 Proofreader

```
100 PRINT"{CLR}PLEASE WAIT...":FORI=886TO
    1018:READA:CK=CK+A:POKEI,A:NEXT
110 IF CK<>17539 THEN PRINT"{DOWN}YOU MAD
    E AN ERROR":PRINT"IN DATA STATEMENTS.
    ":END
120 SYS886:PRINT"{CLR}{2 DOWN}PROOFREADER
     ACTIVATED.":NEW
886 DATA 173,036,003,201,150,208
892 DATA 001,096,141,151,003,173
898 DATA 037,003,141,152,003,169
904 DATA 150,141,036,003,169,003
910 DATA 141,037,003,169,000,133
916 DATA 254,096,032,087,241,133
922 DATA 251,134,252,132,253,008
928 DATA 201,013,240,017,201,032
934 DATA 240,005,024,101,254,133
940 DATA 254,165,251,166,252,164
946 DATA 253,040,096,169,013,032
952 DATA 210,255,165,214,141,251
958 DATA 003,206,251,003,169,000
964 DATA 133,216,169,019,032,210
970 DATA 255,169,018,032,210,255
976 DATA 169,058,032,210,255,166
982 DATA 254,169,000,133,254,172
988 DATA 151,003,192,087,208,006
994 DATA 032,205,189,076,235,003
1000 DATA 032,205,221,169,032,032
1006 DATA 210,255,032,210,255,173
1012 DATA 251,003,133,214,076,173
1018 DATA 003
```

Program 2: Atari Proofreader

```
100 GRAPHICS 0
110 FOR I=1536 TO 1700:READ A:POKE I
    ,A:CK=CK+A:NEXT I
120 IF CK<>19072 THEN ? "Error in DA
    TA statements. Check typing":END

130 A=USR(1536)
140 ? :? "Automatic Proofreader now
    activated."
150 END
1536 DATA 104,160,0,185,26,3
1542 DATA 201,69,240,7,200,200
1548 DATA 192,34,208,243,96,200
1554 DATA 169,74,153,26,3,200
1560 DATA 169,6,153,26,3,162
1566 DATA 0,189,0,228,157,74
1572 DATA 6,232,224,16,208,245
1578 DATA 169,93,141,78,6,169
1584 DATA 6,141,79,6,24,173
1590 DATA 4,228,105,1,141,95
1596 DATA 6,173,5,228,105,0
1602 DATA 141,96,6,169,0,133
1608 DATA 203,96,247,238,125,241
1614 DATA 93,6,244,241,115,241
1620 DATA 124,241,76,205,238,0
1626 DATA 0,0,0,32,62
1632 DATA 246,8,201,155,240,13
1638 DATA 201,32,240,7,72,24
1644 DATA 101,203,133,203,104,40
1650 DATA 96,72,152,72,138,72
1656 DATA 160,0,169,128,145,88
1662 DATA 200,192,40,208,249,165
1668 DATA 203,74,74,74,74,24
1674 DATA 105,161,160,3,145,88
1680 DATA 165,203,41,15,24,105
1686 DATA 161,200,145,88,169,0
1692 DATA 133,203,104,170,104,168
1698 DATA 104,40,96
```

©

How To Type COMPUTE!'s Programs

Many of the programs which are listed in COMPUTE! contain special control characters (cursor control, color keys, inverse video, etc.). To make it easy to tell exactly what to type when entering one of these programs into your computer, we have established the following listing conventions. There is a separate key for each computer. Refer to the appropriate tables when you come across an unusual symbol in a program listing. If you are unsure how to actually enter a control character, consult your computer's manuals.

Atari 400/800

Characters in inverse video will appear like: INVERSE VIDEO
Enter these characters with the Atari logo key, {⅄}.

When you see	Type	See	
(CLEAR)	ESC SHIFT <		Clear Screen
(UP)	ESC CTRL -	↑	Cursor Up
(DOWN)	ESC CTRL =	↓	Cursor Down
(LEFT)	ESC CTRL +	←	Cursor Left
(RIGHT)	ESC CTRL *	→	Cursor Right
(BACK S)	ESC DELETE		Backspace
(DELETE)	ESC CTRL DELETE		Delete character
(INSERT)	ESC CTRL INSERT		Insert character
(DEL LINE)	ESC SHIFT DELETE		Delete line
(INS LINE)	ESC SHIFT INSERT		Insert line
(TAB)	ESC TAB		TAB key
(CLR TAB)	ESC CTRL TAB		Clear tab
(SET TAB)	ESC SHIFT TAB		Set tab stop
(BELL)	ESC CTRL 2		Ring buzzer
(ESC)	ESC ESC		ESCape key

Graphics characters, such as CTRL-T, the ball character ● will appear as the "normal" letter enclosed in braces, e.g. {T}.

A series of identical control characters, such as 10 spaces, three cursor-lefts, or 20 CTRL-R's, will appear as {10 SPACES}, {3 LEFT}, {20 R}, etc. If the character in braces is in inverse video, that character or characters should be entered with the Atari logo key. For example, {♥} means to enter a reverse-field heart with CTRL-comma, {5U} means to enter five inverse-video CTRL-U's.

Commodore PET/CBM/VIC/64

Generally, any PET/CBM/VIC/64 program listings will contain words within braces which spell out any special characters: {DOWN} would mean to press the cursor down key. {5 SPACES} would mean to press the space bar five times.

To indicate that a key should be *shifted* (hold down the SHIFT key while pressing the other key), the key would be underlined in our listings. For example, S would mean to type the S key while holding the shift key. If you find an underlined key enclosed in braces (e.g., {10 N}), you should type the key as many times as indicated (in our example, you would enter ten shifted N's). Some graphics characters are inaccessible from the keyboard on CBM Business models (32N, 8032).

For the VIC and 64, if a key is enclosed in special brackets, ⌊ ⌋, you should hold down the *Commodore key* while pressing the key inside the special brackets. (The Commodore key is the key in the lower left corner of the keyboard.) Again, if the key is preceded by a number, you should press the key as many times as indicated.

Rarely, you'll see in a Commodore 64 program a solitary letter of the alphabet enclosed in braces. These characters can be entered by holding down the CTRL key while typing the letter in the braces. For example, {A} would indicate that you should press CTRL-A.

About the *quote mode*: you know that you can move the cursor around the screen with the CRSR keys. Sometimes a programmer will want to move the cursor under program control. That's why you see all the {LEFT}'s, {HOME}'s, and {BLU}'s in our programs. The only way the computer can tell the difference between direct and programmed cursor control is the quote mode.

Once you press the quote (the double quote, SHIFT-2), you are in the quote mode. If you type something and then try to change it by moving the cursor left, you'll only get a bunch of reverse-video lines. These are the symbols for cursor left. The only editing key that isn't programmable is the DEL key; you can still use DEL to back up and edit the line. Once you type another quote, you are out of quote mode.

You also go into quote mode when you INSerT spaces into a line. In any case, the easiest way to get out of quote mode is to just press RETURN. You'll then be out of quote mode and you can cursor up to the mistyped line and fix it.

Use the following tables when entering special characters:

VIC And 64

When You Read:	Press:	See:	When You Read:	Press:	See:
{CLR}	SHIFT CLR/HOME		{GRN}	CTRL 6	
{HOME}	CLR/HOME		{BLU}	CTRL 7	
{UP}	SHIFT ↑CRSR↓		{YEL}	CTRL 8	
{DOWN}	↑CRSR↓		{F1}	f1	
{LEFT}	SHIFT ←CRSR→		{F2}	f2	
{RIGHT}	←CRSR→		{F3}	f3	
{RVS}	CTRL 9		{F4}	f4	
{OFF}	CTRL 0		{F5}	f5	
{BLK}	CTRL 1		{F6}	f6	
{WHT}	CTRL 2		{F7}	f7	
{RED}	CTRL 3		{F8}	f8	
{CYN}	CTRL 4		←	←	
{PUR}	CTRL 5		↑	SHIFT ↑	

All Commodore Machines

Clear Screen	{CLR}	Cursor Left	{LEFT}
Home Cursor	{HOME}	Insert Character	{INST}
Cursor Up	{UP}	Delete Character	{DEL}
Cursor Down	{DOWN}	Reverse Field On	{RVS}
Cursor Right	{RIGHT}	Reverse Field Off	{OFF}

Apple II / Apple II Plus

All programs are in Applesoft BASIC, unless otherwise stated. Control characters are printed as the "normal" character enclosed in braces, such as {D} for CTRL-D. Hold down CTRL while pressing the control key. You will not see the special character on the screen.

Texas Instruments 99/4

The only special characters used are in PRINT statements to indicate where two or more spaces should be left between words. For example, ENERGY {10 SPACES} MANAGEMENT means that ten spaces should be left between the words ENERGY and MANAGEMENT. Do not type in the braces or the words 10 SPACES. Enter all programs with the ALPHA LOCK on (in the down position). Release the ALPHA LOCK to enter lowercase text.

CAPUTE!

Modifications Or Corrections To Previous Articles

Atari Snertle

Program 3 (p. 94) of this math tutorial from the May issue has a bug in its subtraction routine. In those cases when the answer to the displayed problem should be zero, a zero will not be accepted as the correct result. Donald Carlson points out that line 362 should read as follows:

```
362 IF Q=2 AND K<=L THEN M=L-K
```

64 Hi-Res Graphics Editor

The notes to this program (May issue, p. 82) failed to state the required starting and ending addresses to use when typing the MLX portion of the editor (Program 2). The values are 49152 for the start and 51553 for the end. Also, the series of steps required to set up the program may seem cumbersome. Andy Van Duyne has provided this short program, which will perform all the steps for you:

```
10 IF FL=0 THEN FL=1:LOAD"HIRES/ML",8,1
20 PRINT"{CLR}{2 DOWN}POKE642,128:POKE44,
   128:POKE32768,0:NEW"
30 PRINT"{3 DOWN}LOAD"CHR$(34)"HIRES/BAS"
   CHR$(34)",8"
40 PRINT"{HOME}";
50 POKE 198,6:POKE 631,13:POKE 632,13:POK
   E 633,13
60 POKE 634,82:POKE 635,213:POKE 636,13
```

The program assumes you have used the filenames HIRES/ML for the machine language portion (typed in with MLX) and HIRES/BAS for the BASIC portion (Program 3). Change these names in lines 10 and 30 to match the names you used. To use the program with tape, change the 8 to a 1 in lines 10 and 30. You must have the BASIC portion saved on the tape immediately following the machine language portion.

The screen dump feature of the editor will not work with the new Commodore 1526 printer, since this model does not have the dot-addressable graphics feature of the Commodore 1525 printer. ©

COMPUTER MAIL ORDER

HOME COMPUTERS

ATARI®

1010 Recorder	$74.00	600XL	$189
1020 Color Printer	$249.00	800XL	$299
1025 Dot Matrix Printer	$349.00	1200XL	CALL
1027 Letter Quality Printer	$309.00	1400XL	CALL
1030 Direct Connect Modem	$119.00		
1050 Disk Drive	$399.00		
CX30 Paddle	$12.00		
CX40 Joystick each	$8.00	7101 Entertainer	$69.00
CX77 Touch Tablet	$64.00	7102 Arcade Champ	$75.00
CX80 Trak Ball	$48.00	8026 Dig Dug	$33.00
CX85 Keypad	$105.00	8030 E.T. Phone Home	$33.00
488 Communicator II	$229.00	8031 Donkey Kong	$39.00
4003 Assembler Editor	$47.00	8033 Robotron	$35.00
4011 Star Raiders	$33.00	8034 Pole Position	$39.00
4012 Missile Command	$29.00	8036 Atari Writer	$79.00
4013 Asteroids	$29.00	8040 Donkey Kong, Jr.	$39.00
5049 VisiCalc	$159.00	8043 Ms. Pacman	$39.00
7079 Logo	$79.00	8044 Joust	$39.00

ALIEN VOICE BOX

Atari	$119.00	Apple	$149.00

PRINTERS

AXIOM

AT-100 Atari Interface Printer	$239.00
GP-100 Parallel Interface	$199.00
AT-846 Interface for Atari	$89.00

BMC

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

CENTRONICS

122 Parallel	$299.00
739-1 Parallel	$199.00
739-3 Serial	$249.00

C.ITOH

Gorilla Banana	$209.00
Prowriter 8510P	$379.00
Prowriter 1550P	$599.00
A10 (18 cps)	$569.00
8600 P	$929.00
F10-40	$999.00
F10-55	$1349.00

COMREX

ComWriter II Letter Quality	$499.00

DIABLO

620 Letter Quality	$949.00
630 Letter Quality	$1749.00

DAISYWRITER

2000	$949.00
Tractor Feed	$109.00

EPSON

RX-80, RX-80FT, RX-100	CALL
FX-80, FX-100	CALL
LQ 1500	CALL

IDS

Prism 80...For Configuration	CALL
Prism 32...For Configuration	CALL

JUKI

6100	$499.00

MANNESMAN TALLY

160L	$589.00
180L	$799.00
Spirit 80	$309.00

NEC

8023 Dot Matrix	$379.00
8025 Dot Matrix	$669.00
2010/15/30	$749.00
3510/15/30	$1369.00
7710/15/30	$1799.00

OKIDATA

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

OLYMPIA

Compact 2	$479.00
Compact RO	$509.00
ESW 3000	$1449.00

SMITH CORONA

TP-1000	$399.00
Tractor Feed	$119.00

SILVER REED

500 Letter Quality	$449.00
550 Letter Quality	$569.00
770 Letter Quality	$899.00

STAR

Gemini 10X	$299.00
Gemini 15X	$399.00
Delta 10	$449.00
Serial Board	$75.00
Radix 10	$599.00
Radix 15	$699.00

TOSHIBA

1340	$869.00
1351	$1699.00

TRANSTAR

120P	$499.00
130P	$689.00
315 Color	$499.00

MEMORY BOARDS

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

DISK DRIVES FOR ATARI

PERCOM

AT 88-S1	$369.00
AT 88-A1	$259.00
AT 88-S1 PD	$449.00
RFD 40-S1	$369.00
RFD 40-S2	$699.00
RFD 44-S1	$539.00

INDUS

GT Drive	$379.00

RANA

1000	$329.00

TRAK

AT-D2	$389.00

MODEMS

ANCHOR

Volksmodem	$59.00
Mark IL	$79.00
Mark VII (Auto Ans/Auto Dial)	$119.00
Mark XII (1200 Baud)	$299.00
TRS-80 Color Computer	$99.00
9 Volt Power Supply	$9.00

HAYES

Smartmodem 300	$209.00
Smartmodem 1200	$499.00
Smartmodem 1200B	$449.00
Micromodem IIe	$269.00
Micromodem 100	$299.00
Smart Com II	$89.00
Chronograph	$199.00

NOVATION

J-Cat	$99.99
Cat	$139.00
Smart Cat 103	$179.00
Smart Cat 103/212	$399.00
AutoCat	$219.00
212 AutoCat	$549.00
Apple Cat II	$249.00
212 Apple Cat	$449.00
Apple Cat 212 Upgrade	$269.00

ZENITH

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

DISKETTES

MAXELL

5¼" MD-1	$24.99
5¼" MD-2	$34.99
8" FD-1 (SS/DD)	$39.00
8" FD-2 (DS/DD)	$49.00

VERBATIM

5¼" SS/DD	$26.99
5¼" DS/DD	$36.99

BIB

5¼" Disk Head Cleaner	$14.99

ELEPHANT

5¼" SS/SD	$16.99
5¼" SS/DD	$21.99
5¼" FD/DD	$26.99

DISK HOLDERS
INNOVATIVE CONCEPTS

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

CONTROLLERS & JOYSTICKS

WICO

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

AMIGA

3100 Single	$13.99
3101 Pair	$19.99
Joyboard	$37.99

KRAFT

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

KOALA PADS

Atari (ROM)	$79.00
C-64 (ROM)	$79.00
IBM	$99.00
Apple/Franklin	$85.00

MONITORS

AMDEK

300 Green	$149.00
300 Amber	$159.00
310 Amber	$169.00
Color 1	$279.00
Color 1 Plus	$299.00
Color 2 Plus	$419.00
Color 3	$349.00
Color 4	$749.00

BMC

12" Green	$88.99
12" Green Hi-Res	$119.99
9191-13" Color	$249.00

GORILLA

12" Green	$88.99
12" Amber	$95.99

NEC

JB 1260 Green	$109.00
JB 1201 Green	$149.99
JB 1205 Amber	$159.99
JB 1215 Color	$259.00
JC 1216 RGB	$429.00
JC 1460 Color	$359.00

PRINCETON GRAPHICS

HX-12 RGB	$529.00

SAKATA

100	$269.00

TAXAN

210 Color RGB	$299.00
400 Med-Res RGB	$319.00
415 Hi-Res RGB	$439.00
420 Hi-Res RGB (IBM)	$489.00
100 12" Green	$125.00
105 12" Amber	$135.00

USI

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

QUADRAM

Quadchrome 8400 Color	$529.00

ZENITH

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

east

800-233-8950

In PA call (717)327-9575, Dept. 105
Order Status Number: 327-9576
Customer Service Number: 327-1450
477 E. 3rd St., Williamsport, PA 17701

canada

Ontario/Quebec

800-268-3974

Other Provinces 800-268-4559
In Toronto call (416)828-0866, Dept 105
Order Status Number: 828-0866
2505 Dunwin Drive, Unit 3B
Mississauga, Ontario, Canada L5L1T1

west

800-648-3311

In NV call (702)588-5654, Dept. 105
Order Status Number: 588-5654
P.O.Box 6689
Stateline, NV 89449

No risk, no deposit on C.O.D. orders and no waiting period for certified checks or money orders. Add 3% (minimum $5) shipping and handling on all orders. Larger shipments may require additional charges. NV and PA residents add sales tax. All items subject to availability and price change. Call today for our catalog.
CANADIAN ORDERS: All prices are subject to shipping, tax and currency fluctuations. Call for exact pricing in Canada.

MasterCard

COMPUTER MAIL ORDER

APPLE/FRANKLIN DISK DRIVES

MICRO-SCI

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

RANA

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

APPLE IIe STARTER PACK

64K Apple IIe, Disk Drive & Controller, 80 Column Card, Monitor II & DOS 3.3
COMPLETE .... CALL

FRANKLIN

ACE 1000 Color Computer	CALL
ACE Family Pack System	CALL
ACE PRO PLUS System	CALL
ACE 1200 Office Mgmt. System	CALL

"NOT IIe EXPENSIVE"

MBC 550 .... CALL
MBC 555 .... CALL

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

SANYO

NEC

PC-8201 $489

NEC

PC-8221A Thermal Printer	$149.99
PC-8281A Data Recorder	$99.00
PC-8201-06 8K RAM Chips	$105.00
PC-8206A 32K RAM Cartridge	$329.00

SHARP

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

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

HP 10C	$51.99
HP 11C	$69.99
HP 12C	$88.99
HP 15C	$88.99
HP 16C	$88.99
HP 75C	$749.99
HPIL Module	$98.99
HPIL Cassette or Printer	$359.99
Card Reader	$143.99
Extended Function Module	$63.99
Time Module	$63.99

HEWLETT PACKARD

HP 71B $449.99
41CV $209.99
41CX $249.99

commodore

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

PROFESSIONAL SOFTWARE

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

MSD

SD1 Disk Drive	$349.00
SD2 Disk Drive	$599.00

SX-64 Portable $839

CBM 8032	**$599**
CBM 64	**$199**
C1541 Disk Drive	$249.00
C1530 Datasette	$69.00
C1520 Color Printer/Plotter	$129.00
M-801 Dot Matrix Printer	$219.00
C1526 Dot Matrix/Serial	$299.00
C1702 Color Monitor	$249.00
C1311 Joystick	$4.99
C1312 Paddles	$11.99
C1600 VIC Modem	$59.00
C1650 Auto Modem	$89.00
Logo 64	$49.00
Pilot 64	$39.00
Word Pro 64 Plus	$59.00
Parallel Printer Interface	$49.00
Calc Result 64	$65.00
Calc Result Easy	$39.00
Codewriter 64	$75.00
Quick Brown Fox	$49.00
MCS 801 Color Printer	$499.00
DPS 1101 Daisy Printer	$459.00
Magic Voice Speech Module	$54.00
Desk Organizer Lock	$49.00
Vidtex Telecommunications	$34.95

IBM

NEC PRINTERS

NEC 2050	$899.00
NEC 3550	$1639.00

PERCOM/TANDON DISK DRIVES

5¼" 320K Floppy	$229.00
5 Meg Hard w/Controller	$1049.00
10 Meg Hard w/Controller	$1349.00
15 Meg Hard w/Controller	$1599.00
20 Meg Hard w/Controller	$1899.00

AMDEK

310A Amber Monitor	$169.00
Color 2	$419.00
Color 4	$749.00

AST RESEARCH

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

QUADRAM

Quadlink	$479.00
Quadboard as low as	$289.00
Quad 512 Plus as low as	$249.00
Quadcolor as low as	$219.00
Chronograph	$89.00
Parallel Interface Board	$89.00
64K RAM Chips Kit	$59.00

CONTINENTAL SOFTWARE

1st Class Mail/Form Letter	$79.00
Home Accountant Plus	$88.00

MICROPRO

WordStar Professional Pack	$389.00
InfoStar	$299.00
SpellStar	$159.00
CalcStar	$99.00

MICROSTUF

Crosstalk	$105.00

MICROSOFT

MultiPlan	$159.00

ASHTON-TATE

dBASE II	$389.00
Friday!	$185.00

IUS

EasyWriter II	$249.00
EasySpeller	$119.00
EasyFiler	$229.00

LOTUS

Symphony	CALL
1-2-3	$329.00

PROFESSIONAL SOFTWARE

PC Plus/The Boss	$329.00

SYNAPSE

File Manager	$89.00

SOFTWARE

	C-64	Atari	IBM	Apple
Electronic Arts				
Axis Assassin	$17.95	$17.95	N/A	$17.95
M.U.L.E.	$29.95	$29.95	N/A	N/A
Pinball Construction	$29.00	$29.00	N/A	$29.00
Cut & Paste	$39.95	$39.95	N/A	$39.95
Hard Hat Mack	$29.00	$29.00	N/A	$29.00
InfoCom				
Witness	$32.95	$32.95	N/A	$32.95
Infidel	$32.95	$32.95	$32.95	$32.95
Starcross	N/A	$29.00	$29.00	$29.00
Planetfall	$32.95	$32.95	$32.95	$32.95
Enchanter	$32.95	$32.95	$32.95	$32.95
Zork	N/A	$29.00	$29.00	$29.00
AtariSoft				
Dig Dug	$37.99	N/A	$29.99	$29.99
Defender	$37.99	N/A	$29.99	$29.99
Robotron 2084	$29.99	N/A	$29.99	$29.99
PacMan	$37.99	N/A	$29.99	$29.99
Donkey Kong	$37.99	N/A	$29.99	$29.99
Spinnaker				
Story Machine	$24.00	$24.00	$29.00	$29.00
Cosmic Life	$24.00	$24.00	N/A	N/A
Up for Grabs	$29.00	$29.00	N/A	N/A
Kindercomp	$21.00	$21.00	$24.00	$21.00
Snooper Troops	$29.00	$29.00	N/A	$29.00
Delta Drawing	$29.00	$29.00	$39.00	$39.00
Kids on Keys	$29.00	$29.00	N/A	N/A
Hey Diddle Diddle	$21.00	$21.00	$21.00	$21.00
Facemaker	$24.00	$24.00	$24.00	$24.00
Rhymes & Riddles	N/A	$21.00	$24.00	$24.00
Dynatech				
Code Writer	$79.00	$79.00	$175.00	$155.00
VisiCorp				
VisiCalc	$159.00	$159.00	$159.00	$159.00
VisiCalc Advanced	N/A	N/A	B/A	$269.00
Visitrend/Plot	N/A	N/A	$199.00	$199.00
VisiFile	N/A	N/A	$199.00	$169.00
VisiSchedule	N/A	N/A	$199.00	$199.00
Desktop Plan	N/A	N/A	$199.00	$169.00
Business Forecast Model	N/A	N/A	$75.00	$75.00
StretchCalc	N/A	N/A	$75.00	$75.00
VisiTutor Calc	N/A	N/A	$59.00	$59.00
VisiTutor Advanced	N/A	N/A	$75.00	$75.00
VisiTutor Word	N/A	N/A	$59.00	$59.00
pfs:				
Write	N/A	N/A	$89.00	$79.00
Graph	N/A	N/A	$89.00	$79.00
Report	N/A	N/A	$79.00	$79.00
File	N/A	N/A	$89.00	$79.00
Solutions:* as low as	N/A	N/A	$16.00	$16.00

*Call on Titles

east
800-233-8950
In PA call (717)327-9575, Dept. 105
Order Status Number: 327-9576
Customer Service Number: 327-1450
477 E. 3rd St., Williamsport, PA 17701

canada
Ontario/Quebec
800-268-3974
Other Provinces 800-268-4559
In Toronto call (416)828-0866, Dept. 105
Order Status Number: 828-0866
2505 Dunwin Drive, Unit 3B
Mississauga, Ontario, Canada L5L1T1

west
800-648-3311
In NV call (702)588-5654, Dept. 105
Order Status Number: 588-5654
P.O.Box 6689
Stateline, NV 89449

INTERNATIONAL ORDERS: All shipments outside the Continental United States must be pre-paid by certified check only. Include 3% (minimum $5) shipping and handling.
EDUCATIONAL DISCOUNTS: Additional discounts are available to qualified Educational Institutions.
APO & FPO: Add 3% (minimum $5) shipping and handling.

VISA

Lyco Computer Marketing & Consultants

TO ORDER CALL US TOLL FREE **800-233-8760**
In PA 1-717-327-1824

DISK DRIVES

RANA
1000 ... $ CALL

PERCOM
ATARI AT88-PD $329.00
AT-88 DOUBLER $129.00
ADD-ON DRIVES .... $ CALL

HARD	FOR	5 MEG
DISK	APPLE	10 MEG
DRIVES	IBM-PC	15 MEG
AVAILABLE		20 MEG

TRAK DISK DRIVES
AT-D1 .... $379.00
AT-D2 .... $399.00
AT-D2 TURBO PAK $22.95
PRINTER CABLE $22.95

CONCORD DISK DRIVES
APPLE 163K DRIVE $199.00
APPLE 326K DRIVE $229.00
APPLE CONTROLLER CARD $69.00
COMMODORE VIC 174K ... $289.00
COMMODORE VIC 348K ... $359.00
ATARI 176K MASTER ... $289.00
ATARI 348K MASTER ... $369.00
ATARI ADD-ON DRIVES ... $CALL

commodore

VIC 64 .... $ CALL
VIC 1541 .... $239.00

SSI
KNIGHTS OF DESERT ... $26.75
EAGLES $26.75
TIGERS IN SNOW $26.75
COMBAT LEADER $26.75
BATTLE FOR N $26.75

EPYX 64
JUMPMAN C/D $27.75
JUMPMAN JR R $27.75
PITSTOP R $27.75
TEMPLE ASPHAI $27.75
GATE T ASPHAI $27.75
CRUSH, C&C C/D $21.75

INFOCOM 64
ENCHANTER $34.75
PLANETFALL $34.75

DATASOFT
POOYAN C/D $21.75
O'RILEYS MINE $21.75

PARKER 20
FROGGER $33.75

PROFESSIONAL SOFTWARE
WORD PRO 3 .... $59.75
SPELL RIGHT $39.75
BOTH TOGETHER $79.75

EASTERN HOUSE
RABBIT 20 ROM $34.75
RABBIT 64 ROM $34.75

DISKETTES

ELEPHANT
SINGLE SIDE SD (10) $17.75
SD (100) $16.75/10
SINGLE SIDE DD (10) $21.75
DD (100) $20.75/10
DOUBLE SIDE DD (10) ... $26.75
DD (100) $24.75/10

MAXELL
MD1 (10) $27.75
MD2 (10) $37.75

CERTRON CASSETTES
CC-10 (12) $15.99
CC-20 (12) $17.99

COMPUTER CARE

BIB
DISK DRIVE CLEANER ... $12.75
COMPUTER CARE KIT ... $19.75

NORTRONICS
DISK DRIVE CLEANER WITH SOFTWARE FOR IBM-PC, ATARI, VIC, APPLE, & TI $29.75
DISK CLEANER REFILLS ... $14.75
CASS DRIVE CLEANER $9.95
MEDIA BULK ERASER ... $46.75

INNOVATIVE CONCEPTS

DISK STORAGE (10) $4.95
DISK STORAGE (15) $9.95
DISK STORAGE (25) $19.95
DISK STORAGE (50) $26.75
ROM STORAGE (10) $24.75

MODEMS

ANCHOR MARK I $45.75
ANCHOR MARK II $79.75
HAYES SMART $239.00
HAYES MICRO II $309.00
MICROBIT 1000C $129.00
CAT $144.00
D-CAT $155.00
J-CAT $115.00
APPLE CAT II $279.00
212 APPLE CAT $589.00

MONITORS

SAKATA COLOR $249.00
TAXAN GREEN $119.00
TAXAN AMBER $129.00
TAXAN RGB COMPOSITE $289.00
ZENITH AMBER $105.00
ZENITH GREEN $95.00
GORILLA GREEN $88.00
GORILLA AMBER $95.00
NEC JB1260 .... $99.00
NEC JB1205 .... $145.00
NEC JB1215 COLOR ... $269.00
AMDEK GREEN $145.00
AMDEK AMBER $149.00
AMDEK COLOR 1 .... $289.00

apple

BRODERBUND
BANK STREET D $44.75
CHOPLIFTER D $24.95
DAVID'D MAGIC $24.95
DROL D $24.95
LODE RUN D $24.95
SEAFOX D $22.95
SPARE CHANGE D $24.95
AE D $24.75

KOALA
KOALA TABLET $84.75
PROGRAMMERS GUIDE ... $12.75
GEOMETRIC DESIGNS ... $22.75
LOGO DESIGN $27.75
ILLUSTRATOR $99.75
SONG WRITER D $27.75
MASTER TYPE $27.75

CONTINENTAL
HOME ACCOUNT D $44.75
TAX ADVANTAGE $35.75
BOOK OF APPLE SOFTWARE $16.95
GENERAL LEDGER $179.95
ACCOUNTS PAY $179.95
ACCOUNTS REC $179.95
PAYROLL $179.95
PROPERTY MGMT $329.95

INFOCOM
DEADLINE $34.75
ZORK 1,2, or 3 .... $34.75
ENCHANTER $34.75
PLANETFALL $34.75

SPINNAKER
KINDERCOMP D $21.75
STORY MACH D $23.75
FACE MAKER D $23.75
SNOOPER TR D $29.75
SNOOPER T2 D $29.75
DELTA DRAW D $32.75
FRACTION F D $23.75
ALPHABET ZOO D $21.75
MOST AMAZING D $26.75
RHYMES & RID D $21.75

APPLE DUMPLING GX $99.75
APPLE DUMPLING 16K BUFFER $179.75
TEXT PRINTER INTERFACE $79.75

SSI
KNIGHTS OF DESERT ... $26.75
EAGLES $26.75
TIGERS IN SNOW $26.75
GERMANY 1985 .... $36.75
BATTLE FOR NORMANDY $26.75
SHATTERED ALLIANCE ... $39.75

SIERRA ON-LINE
ULTIMA II $39.75
DARK CRYSTAL $27.75

ATARISOFT
PACMAN $25.75
DONKEY KONG $25.75
DIG DUG $25.75
CENTIPEDE $25.75
ROBOTRON $25.75

PROGRAM DESIGN
ANALOGIES $18.50
PREP FOR SAT'S $79.75
PRESCHOOL IQ BUILDER $18.75
READING COMP $18.75
VOCABULARY BUILDER ... $18.75

PEACHTREE SOFTWARE

LIST MANAGER $199.00
BUSINESS GRAPHICS $225.00
GRAPHICS LANGUAGE ... $299.00

REQUIRES COBAL RUNTIME

ACCTS PAYABLE I $495.00
ACCTS PAYABLE II $1695.00
PAYROLL I $495.00
PAYROLL II $1695.00
GENERAL LEDGER I $495.00
GENERAL LEDGER II ... $1695.00

CP/M VERSION
PEACHPACK 4 .... $295.00
ACCTS PAYABLE III $549.00
ACCTS PAYABLE IV $899.00
ACCTS REC III $549.00
ACCTS REC IV $899.00
GENERAL LEDGER III ... $549.00
GENERAL LEDGER IV ... $899.00
INVENTORY MGMT I $549.00
INVENTORY MGMT II ... $899.00
PAYROLL III $549.00
PAYROLL IV $899.00
SALES INVOICING I $549.00
SALES INVOICING $899.00

ATARI

INFOCOM
DEADLINE $34.75
ZORK 1,2, or 3 .... $26.75
ENCHANTER $34.75

EDUFUN
GULP ARROW GRAPHICS $24.75
FACE FLASH $24.75

CONTINENTAL
HOME ACCOUNT D $44.75
TAX ADVANTAGE $35.75
BOOK OF ATARI SOFTWARE $16.95

SCARBOROUGH
SONG WRITER D $27.75
MASTER TYPE $27.75

EASTERN HOUSE
MONKEY WRENCH II $52.75

DON'T ASK SOFTWARE
SAM $39.75
ABUSE $22.75
WORD RACE $24.75

PROGRAM DESIGN
ANALOGIES $14.50
PREP FOR SAT'S $79.75
PRESCHOOL IQ BUILDER $16.75
READING COMP $16.75
VOCABULARY BUILDER ... $16.75
LET'S SPELL $14.75

MasterCard **TO ORDER** VISA

CALL TOLL FREE
800-233-8760
In PA 1-717-327-1824

or send order to
Lyco Computer
P.O. Box 5088
Jersey Shore, PA 17740

Lyco Computer Marketing & Consultants

TO ORDER CALL US TOLL FREE **800-233-8760**
In PA 1 717-327-1824

SAVE ON THESE IN-STOCK PRINTERS

MANNESMANN TALLY

SPIRIT 80 .... $299.00
MTL-160L $559.00
MTL-180L $775.00

NEC

NEC8023 .... $369.00
NEC8025 .... $699.00

PRINTER INTERFACING

AVAILABLE FOR COMMODORE VIC, APPLE, ATARI, IBM-PC, TRS-80, TI, AND OTHERS

STAR MICRONTICS

GEMINI 10X $259.00
GEMINI 15X $379.00
DELTA 10 .... $449.00
DELTA 15 .... $525.00
RADIX 10 .... $575.00
RADIX 15 .... $675.00
POWERTYPE .... $CALL
SWEET P (Model 100) ... $549.00
STX80 .... $149.00

SMITH CORONA TP2 ... $449.00

CITOH

GORILLA GX-100 .... $169.00
GORILLA SERIAL $199.00
PROWRITER 8510 .... $335.00
PROWRITER II $575.00
8600 .... $899.00
STARWRITER $999.00
PRINTMASTER $1299.00
SHEET FEEDER $425.00
620 .... $929.00
630 .... $1699.00
8510 SP $499.00
8510 SCP $559.00
A10 LETTER QUAL $499.00

EPSON

RX-80 .... $CALL
RX-80 FT .... $FOR
FX-80 .... $CURRENT
FX-100 .... $PRICES

OKIDATA

80 .... $SAVE
82A .... $ON THESE
83A .... $IN-STOCK
84 .... $PRINTERS
92 .... $CALL &
93 .... $SAVES

PRINTING PAPER

3000 SHEETS FANFOLD $42.75
1000 SHEETS FANFOLD $19.75
1000 SHEETS LETTER ... $21.95
200 SHEETS LETTER $8.99
150 RAG STATIONERY ... $10.99
MAILING LABELS (1 in) ... $9.75
14x11 1000 FANFOLD ... $24.75

WICO COMMODORE & ATARI

JOYSTICK $16.75
RED BALL $18.75
BOSS JOYSTICK $12.75
TRACK BALL $32.75
12 FT EXT CORD $6.95
TI ADAPTER $9.95
APPLE ANALOG $34.75
IBM-PC ANALOG $34.75

CONTINENTAL 64

HOME ACCOUNT D $44.75
TAX ADVANTAGE $35.75
F.C.M. $35.75
64 USERS OF ENCYC $12.75
20 ENCYCLOPEDIA $12.75

KOALA 64

KOALA TABLET $69.75
PROGRAMMERS GUIDE ... $12.75
GEOMETRIC DESIGNS ... $22.75
LOGO DESIGN $27.75
SPIDER EATER $22.75

commodore

CARDCO

G PRINTER INTERFACE $64.75
B PRINTER INTERFACE $39.75
KEY PAD 64 .... $29.75
LIGHT PEN $29.75
3 SLOT EXPANSION $24.75
5 SLOT EXPANSION $54.75
6 SLOT EXPANSION $79.75
PRINTER UTILITY $19.75
CASSETTE INTERFACE ... $29.75
WRITE NOW 20 .... $29.75
WRITE NOW 64 .... $39.75
MAIL NOW $29.75
DM-1 PRINTER $109.75
LQ-1 PRINTER $499.75

BRODERBUND 64

BANK STREET D $44.75
CHOPLIFTER R $29.95
DAVID'D MAGIC $24.95
DROL D/R $24.95
LODE RUN D/R $24.95
SEAFOX R $29.95
SPARE CHANGE D $24.95

DUST COVERS

800 .... $3.99
400 .... $3.99
1200 .... $3.99
410 .... $3.99
810 .... $3.99
600 XL $5.99
800XL $5.99
1050 .... $5.99
1010 .... $5.99
PROWRITER $5.99
GEMINI 10X $5.99
PERCOM $5.99
EPSON $5.99
RANA $5.99
VIC 64/20 .... $5.99
VIC 1541 .... $5.99

RESTON 64

MINER 2049 ROM . $29.75

TIMEWORKS

INVENTORY $39.75
ACCOUNTS REC ... $39.75
ACCOUNTS PAY ... $39.75
GENERAL LEDGER . $39.75
PAYROLL $39.75
INVENTORY MAN .. $39.75
CASH FLOW $39.75
SALES ANALYSIS .. $39.75
DATA MANAGER ... $19.75
MONEY MGR C/D ... $19.95
CHECKBOOK C/D .. $19.95
FORGET-ME-NOT .. $19.95
STAR BATTLE $19.95
ROBOT REVOLT $19.95
SPELLBOUND C/D . $19.95

SCARBOROUGH 64

SONG WRITER D ... $27.75
PHI BETA FILER $32.75
MASTER TYPE $27.75

HES 64

TURTLE TUTOR $22.75
TURTLE TRAINER .. $22.75
TURTLE GRAPHICS $37.75
SOUND BOX $9.95
64 FORTH $39.95
HESMON $22.75
GRIDRUNNER $19.75
ATTACK OF M C $22.75
HESWRITER $28.75
OMNI WRITER $45.75
TYPE N WRITER $24.75
PAINT BRUSH $22.75
BENJI $25.75
HOME MANAGER .. $28.75
TIME MONEY MGR . $44.75
OMNI CALC $33.75
SWORD POINT $19.95
HES MODEM $49.95
M. MULTIPLAN $65.75

SYNAPSE 64

BLUE MAX C/D $24.75
ZEPPELIN C/D $24.75
PHAROAH'S C/D $24.75
SHAMUS C/D $24.75

PARKER 64

FROGGER $32.75
QBERT $32.75
TUTANKHAM $32.75

SPINNAKER 64

KINDERCOMP D/R $21.75
STORY MACH ROM $24.75
FACE MAKER D/R $24.75
SNOOPER TR D $26.75
SNOOPER T2 D $26.75
DELTA DRAW ROM $26.75
FRACTION F ROM $24.75
KIDS ON KEYS $24.75
ALPHABET ZOO $24.75
MOST AMAZING R $26.75

ADVENTURE INTERNATIONAL

ADVENTURE 1-12 ea. $17.75
SAGA $27.75
PREPPIE $24.75
DISKEY $34.75
SEA DRAGON $24.75

KOALA TABLET $69.75
PROGRAMMERS G ... $12.75
GEOMETRIC DESIGN . $22.75
LOGO DESIGN $27.75
ILLUSTRATOR $99.75

INHOME

400 KEYBOARD .. $IN-STOCK

BIT 3

80 COLUMN SCREEN BOARD $259.75

PARKER

FROGGER $32.75
QBERT $32.75

SPINNAKER

KINDERCOMP D/R .. $20.75
STORY MACH D/R ... $26.75

PERCOM

AT88S1 .. $249.00
RDF40SI $269.00

SSI

KNIGHTS OF DESERT $26.75
EAGLES $26.75
TIGERS IN SNOW $26.75
COMBAT LEADER $26.75

DATASOFT

SPELL WIZZARD $34.75
TEXT WIZZARD $34.75
WORD WIZZARD $46.75
ZAXXON C/D $26.75

GEMINI 10 .... $259.00

BRODERBUND

BANK STREET D ... $44.75
CHOPLIFTER D $32.95
DAVID'D MAGIC $24.95
DROL D $24.95
LODE RUN D $24.95
SEAFOX D $22.95
SPARE CHANGE D . $24.95
AE D $24.75

EPYX

JUMPMAN C/D $27.75
JUMPMAN JR R $27.75
PITSTOP R $27.75
TEMPLE ASPHAI ... $27.75
GATE T ASPHAI $27.75
CRUSH, C&C C/D ... $21.75

LJK

LETTER PERFECT .. $69.75
DATA PERFECT $69.75
SPELL PERFECT ... $69.75

ATARI

PACMAN $29.75
DONKEY KONG $29.75
DIG DUG $29.75
CENTIPEDE ... $29.75
STAR RAIDERS $32.75
ROBOTRON $32.75
PAINT $32.75
JOUST $35.75
LOGO $69.75
ATARI WRITER $79.75
BOOK KEEPER $104.75
VISICALC $159.75
HOME FILING MGR $35.75
PILOT HOME $54.75
KEYPAD $95.75

ATARI Computers for people.

800XL .... $ CALL
1050 DISK DRIVE $ CALL
1050 RECORDER $74.00
1027 PRINTER $ CALL
1025 PRINTER IN-STOCK

MasterCard VISA

TO ORDER CALL TOLL FREE
800-233-8760
Customer Service 1-717-327-1825

or send order to
Lyco Computer
P O Box 5088
Jersey Shore PA 1774C

POLICY

In-stock items shipped within 24 hours of order. Personal checks require four weeks clearance before shipping. No deposit on C.O.D. orders. Free shipping on prepaid cash orders within the continental U.S. PA residents add sales tax. All products subject to availability and price change. Advertised prices show 4% discount offered for cash, add 4% for Master Card or Visa. DEALER INQUIRIES INVITED.

SAVE ON MEMOREX DISKETTES

$1.80 ea. 5¼" SSDD Qty. 20 **$2.35** ea. 5¼" DSDD Qty. 20

5¼" SSDD-96TPI **$2.49** ea. 5¼" DSDD-96TPI **$3.25** ea.
Boxed in 10's with Tyvec sleeves, reinforced hubs and labels.

DISKETTES FOR MACINTOSH & HP-150 AT TERRIFIC SAVINGS!
3½" Memorex .... **$3.99** ea.

See our big ad in this issue for other great values!

Shipping: 5¼" or 3½" DISKETTES—Add $3.00 per 100 or fewer diskettes. **Payment:** VISA and Mastercard accepted. COD orders only, add $3.00 handling charge. **Taxes:** Illinois residents, please add 8% sales tax.

WE WILL BEAT ANY NATIONALLY ADVERTISED PRICE ON THE SAME PRODUCTS AND QUANTITIES!

Nationwide: 1-800-621-6827
Illinois: 1-312-944-2788
Hours: 9AM - 5PM Central Time
Minimum Order: $35.00

DISK WORLD!, Inc.
Suite 4806 • 30 East Huron Street • Chicago, Illinois 60611

MEMOREX

BREAK-THRU

ONLY **$34.95**
SHIPPING & HANDLING ADD $2.50
Dealer inquiries welcome

A compact and inexpensive Eprom eraser for the hacker. It erases two chips per exposure, so if you are one of those smart people who only makes little mistakes and only needs to erase two Eproms at a time, this eraser is for you.

WALLING CO.
7755 E. Evans • Suite 400 • Scottsdale, AZ 85260
(602) 998-7550
Arizona residents add 6% tax

Aspen Ribbons® brand replacements for

Tally® MT80 "Spirit" Microprinter Ribbons

Buy directly from a manufacturer and save! These ribbons are manufactured by Aspen Ribbons, Inc., as replacement ribbons for use on printers manufactured by other companies.* Standard ink color is black. Red, green, blue, brown, and purple colors are available for $2.00 extra per ribbon.

CALL FOR FREE CATALOG.

PRICES (BLACK NYLON)

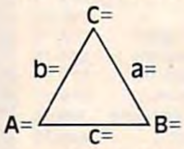
Tally® "Spirit" $4.25 to $7.75 ea.
Price depends on quantity ordered.

*Aspen Ribbons, Inc., is not affiliated with or licensed by any other company.

Aspen Ribbons, Inc. • 1700 N. 55th St.
Boulder, CO 80301-2796 • (303)444-4054
Telex: 45-0055 • End User: 800-525-0646
Wholesale: 800-525-9966

C-64™ and ATARI 400™ OWNERS

SOLVE YOUR TRIGONOMETRY PROBLEMS THE EASY WAY AND SEE RESULTS



$10.00 (DISK ONLY)

Check, Money Orders, VISA and MASTERCARD accepted.
Please specify C-64™ or Atari-400™

Just Fun Software
P.O. Box 25854
Colorado Springs, CO 80917
(303) 597-1965

C-64 is a trademark of Commodore Electronics, Ltd.
Atari is a trademark of Atari Corporation.

SAVE MORE THAN EVER ON 3M Scotch® DISKETTES

LIFETIME WARRANTY!

$1.85 ea. 5¼" SSDD ← (744) Qty. 20
$2.35 ea. 5¼" DSDD (745) → Qty. 20

5¼" SSQD—96TPI (746)	**$2.60** ea.
5¼" DSQD—96TPI (747)	**$3.25** ea.
8" SSSD (740)	**$2.05** ea.
8" SSDD (741)	**$2.50** ea.
8" DSDD (743)	**$3.10** ea.

Shipping: 5¼" DISKETTES—Add $3.00 per 100 or fraction thereof. 8" DISKETTES—Add $4.00 per 100 or fraction thereof. OTHER ITEMS: Shipping charges as shown in addition to diskette shipping charges. **Payment:** VISA or MasterCard. COD orders only, add $3.00. **Taxes:** Illinois customers, please add 8%.

Hours: 9 AM-5 PM Central Time
For fast service call
Nationwide: 1-800-621-6827
In Illinois: 312-944-2788

DISK WORLD!
Suite 4806 • 30 East Huron Street • Chicago, Illinois 60611

Authorized Distributor Information Processing Products 3M

SU Software Unlimited

KOALA SALE!

Koala Pads	
Atari, C-64, Vic-20 w/Disk Software	*$65.00*
Atari, C-64 w/Catridge Software	*$73.00*
Apple II, IIe II+	*$88.00*
IBM P.C.	*$108.00*
Instant Programmers Guide All Computers	*$12.00*
THE Clone Machine	
with "Unguard" C-64	*$32.00*
S.A.M., A. C. Disk	*$39.95*
Chatterbee A. C. Disk	*$27.00*

SOFTWARE UNLIMITED, 27 LOOKOVER LN., YARDLEY, PA 19067. VISA & MC ADD 4%. PLEASE ADD $1.50 P&H. PERSONAL CHECKS REQUIRE 2 WEEKS TO CLEAR. CALL FOR A COMPLETE LIST OF OUR SOFT & HARDWARE.
(215) 493-1372
OUT OF STATE 1-800-225-3656

WHERE DID THAT *@#% VALUE COME FROM?

Xref HELPER™

from

HELPS YOU FIND OUT!

From your Commodore 64 BASIC program on diskette, Xref HELPER sorts and lists the variables, constants, GOTOs, and GOSUBs with the line number of each occurrence. Fast, Xref HELPER scans most programs in one to five minutes. Printer required.

VISA $25.00 plus $1.25 shipping MasterCard
MN residents please add 6% sales tax
Canadian residents may pay CDN $32.40 plus CDN $1.60 shipping

(M)agreeable software, inc.
5925 Magnolia Lane • Plymouth, MN 55442
(612) 559-1108

HELPER is a trademark of (M)agreeable Software, Inc.
Commodore is a trademark of Commodore Electronics Ltd.

DISKETTES

$1.35 100% Certified

Buy 100 Diskettes GET 10 FREE

DISKMINDER™ Capacity 75 $17.95

1-800-322-DATA

GREAT DISKETTES Super low prices SYNCOM

The low priced, high quality diskette with a LIFETIME WARRANTY. Packed in polybags of 10 with Tyvek envelopes, labels and reinforced hubs. One of the best buys we've seen.

$1.39 ◀ ea. 5¼" SSDD QTY. 20
$1.85 5¼" DSDD ea. ▶ QTY. 20

OTHER GREAT VALUES:

DISKETTE 70—Holds 70 5¼" diskettes in dust free safety **$14.95** ea. + $3.00 Shpng.
DISK CADDIES—Flip up style holds 10 5¼" diskettes **$1.65** ea. + .20 Shpng.

$15.00 Bonus Offer!

For the lowest priced, highest quality diskettes, storage cases, printer ribbons and paper products, send for our catalog. FREE with your order; $1.00 otherwise. BONUS! Every catalog includes $15.00 worth of bonus coupons.

Shipping: 5¼" DISKETTES—Add $3.00 per 100 or fraction thereof. OTHER ITEMS: Shipping charges as shown in addition to diskette shipping charges. **Payment:** VISA or MC. COD orders, add $3.00. **Taxes:** Illinois customers, please add 8%.

Nationwide: 1-800-621-6827
In Illinois: 1-312-944-2788
Minimum Order: $35.00
WE WILL BEAT ANY NATIONALLY ADVERTISED PRICE!
DISK WORLD!
Suite 4806 • 30 East Huron Street • Chicago, Illinois 60611

SYNCOM

VIDEO & COMPUTER REVOLUTION

BSD ENTERPRISES

800-346-5001 OR 212-438-4111

9 AM TILL 9 PM
800 346 5001
212 438 4111
SUNDAY THRU FRIDAY

ALL PRODUCTS ARE IN ALPHABETICAL ORDER. WE CARRY A **FULL** LINE OF LISTED BRANDS.
IF YOU DO NOT SEE WHAT YOU WANT, CALL AND ASK FOR IT.

WHOLESALERS, DEALERS AND OTHER DISTRIBUTORS INQUIRIES INVITED...
212-438-4111 ext. #18

COMPUTERS

ALTOS
586-20 5599.95
586-40 6999.95
986-40 8399.95

ATARI
600XL 135.95
800XL 234.95
1200XL *CALL
1400XL *CALL
1450XLD *CALL

APPLE
IIe Single D.D. *CALL
IIe Duo D.D. *CALL
III *CALL
LISA *CALL
MACINTOSH *CALL

COLECO
ADAM 524.95

COLUMBIA
1600 1V 2249.95
1600 4V 3399.95
1600 VP 2149.95

COMMODORE
VIC 20 75.95
C-64 175.95
EXEC. 64 679.95

COMPAQ
COMPAQ *CALL
COMPAQ + *CALL

CROMEMCO
FULL LINE *CALL

EAGLE
SPIRIT 2 2299.95
SPIRIT XL 3299.95
IIE-1 1435.95
IIE-2 1799.95
IIE-3 2199.95
IIE-4 3275.95
PC-1 2019.95
PC-2 2349.95
PC-XL 3059.95
1620 3039.95
1630 4349.95
1640 5599.95

EPSON
2X-10 *CALL

FRANKLIN
ACE 1000 *CALL
ACE FAM. PACK *CALL
ACE PRO + *CALL
ACE 1200 OMS *CALL

HEWLETT-PACKARD
41C 129.95
41CV 189.95
41CX 225.95
71B 369.95
75C 689.95
150 2899.95

I.B.M.
PC *CALL
PC jr *CALL

I.B.M.
PC *CALL
PC JR *CALL
PC-XT *CALL

KPYPRO
II *CALL
IV *CALL
10 *CALL

NEC
PC8201A 539.95
H01 1869.95
H02 2344.95
H03 2854.95
H04 2378.95

OLIVETTI
M18 DT2 *CALL
M18-PT *CALL

RADIO SHACK
100 664.95
4 829.95
4P 1499.95
12 2324.95
16B 3299.95
2000 2299.95

SANYO
550 *CALL
555 *CALL
1100 1199.95
1150 1479.95
1160 1749.95
1200 1399.95
1250 1819.95
4000 1539.95
4050 1889.95

SHARP
PC 1250 75.95
PC 1500 135.95
PC 5000 *CALL

TELEVIDEO
TELEPORT PORT *CALL
800A 999.95
802 2449.95
803 1799.95
802H 4299.95
806/20 4749.95
816/40 8799.95
1602 *CALL
1603 *CALL

TEXAS INSTRUMENTS
PRO 1669.95

TIMEX-SINCLAIR
1000 21.95
2068 *CALL

DISK DRIVES

AMDEK
5¼" 199.95

ATARI
1050 279.95

COMMODORE
154 179.95

COMREX
10 MB *CALL

CORVUS
6 MB 1559.95
11MB 2125.95
20 MB 2799.95

DAVONG
10 MB 1499.95
15 MB 1929.95
21 MB 2199.95

EAGLE
320 KB floppy 349.95
10 MB 1749.95
32 MB 2799.95

FRANKLIN
ACE 2 *CALL

MICRO-SC1
A-2 169.95
XL 139.95
A-82 359.95

PEGASUS
10 MB 1069.95
23 MB 1599.95

PERCOM
AT 88-S1 279.95
AT 88-S1PD 365.95
AT 88-A1 219.95
RFD 40-S1 345.95
RFD 40-S2 569.95
RFD 40-A1 219.95
RFD 44-S1 399.95
RFD 44-S2 679.95
5 MB 1299.95
10 MB 1499.95
15 MB 1899.95
20 MB 2229.95

QUADRAM
QUADISK-ENTIRE LINE *CALL

RANA
1000 259.95
ELITE I 239.95
ELITE II 389.95
ELITE III 499.95
FOR I.B.M. 239.95

SANYO
3200 *CALL
6400 *CALL
EDD & EFD *CALL

TALLGRASS
12 *CALL
20 *CALL

TECMAR
10 MB 1995.95
15 MB 2299.95
33 MB 3099.95

XEROX
5¼" DUAL 569.95
8" DUAL 1049.95

MODEMS

ANCHOR
SIGNALMAN 59.95
MARK I 59.95
MARK II 59.95
MARK III 79.95
MARK V 79.95
MARK VI 159.95
MARK VII 89.95
MARK XII 239.95

CHARACTER GENERATOR

Attention all VIC 20 owners, now you yes you can put any and all titles you want in any place in any color on your video tape for only

$79.95

HAYES
CHRONOGRAPH 169.95
SMARTCOM I 59.95
SMARTCOM II 74.95
MICROMODEM II 209.95
M/M II + 225.95
M/M IIE 209.95
M/M 100 259.95
S/M 300 199.95
S/M 1200 449.95
S/M 1200B 399.95

NOVATION
CAT 119.95
J-CAT 85.95
SMART CAT 103 145.95
SMART CAT 103/212 345.95
212 AUTO CAT 509.95
APPLE-CAT II 229.95
212 APPLE CAT 519.95
212 APPLE CAT UPGRADE 289.95
ACCESS 1-2-3- 369.95
D-CAT 129.95
EXEC. INFONE 699.95
MAXI INFONE 849.95

TRANSEND
TRANSPAK I 265.95
TRANSPAK II 299.95
TRANSPAK III 369.95
P.C. MODEM 235.95
P.C. TRANSPAK 324.95

U.S. ROBOTICS
PASSWORD 299.95
AUTO DIAL 399.95

MONITORS

AMDEK
12" GR. 109.95
12" AMB. 119.95
13" COL. 249.95
13" COL. RGB 369.95

BMC
12" GR 69.95
12" AMB 72.95
13" COL. (9191) 199.95

COMMODORE
170 189.95

HITACHI
CM 1481 *CALL

EAGLE
12" GR. 229.95
14" COL. RGB 479.95

GORILLA
12" GR. 69.95
12" AMB 79.95

NEC
9" GR 119.95
12" GR. 89.95
12" AMB. 135.95
12" COL. 249.95
12" COL. RGB 389.95

PANASONIC
12" GR
12" GR. 89.95
12" AMB 135.95
12" COL 249.95
12" COL. RGB 389.95

PANASONIC
12" GR 129.95
12" AMB. 139.95
10" COL. 219.95
13" COL. RGB 309.95

PRINCETON GRAPHICS
12" COL. RGB 469.95

SAKATA
12" GR. *CALL
13" COL. *CALL

QUADRAM
QUADCHROME 459.95
QUADCOLOR I 179.95

SANYO
ENTIRE LINE *CALL

TAXAN
12" GR. 99.95
12" AMB. 109.95
12" COL. 239.95
12" COL. RGB 259.95

USI
FULL LINE *CALL

ZENITH
FULL LINE *CALL

PLOTTERS

AMDEK
DXY 100 519.95
AMPLOT II *CALL

ENTER
SWEET-P 419.95
SWEET-P GRAPH. PKG. 669.95
SWEET-P PRIME 619.95

HEWLETT-PACKARD
7470 A 769.95
7475 1389.95

MANNESMAN TALLY
PIXY-3 *CALL

PANASONIC
VP 6801-A 1179.95

BMC
X-Y 749.95

PRINTERS

ATARI
1020 189.95
1025 269.95
1027 246.95

ATARI
1020 189.95
1025 269.95
1027 246.95

AXIOM
ENTIRE LINE *CALL

BROTHER
HR 15 389.95
HR 25 589.95

BMC
BX 80 199.95
PB 401 569.95
BX 180 249.95

C. ITOH
PROWRITER 299.95
STARWRITER 849.95
PRINTMASTER 1139.95

CENTRONICS
ENTIRE LINE *CALL

COMMODORE
1526 225.95
MPS-801 179.95

ANADEX
ENTIRE LINE *CALL

COMREX
ENTIRE LINE *CALL

DAISYWRITER
1500 739.95
2000 879.95

DATA PRODUCTS/IDS
PRISM 80 899.95
REST OF LINE AND CONFIGURATIONS *CALL

DIABLO
620 819.95
630 1559.95

EPSON
ENTIRE LINE *CALL

GORILLA
BANANA 129.95

JUKI
6100 389.95

NEC
3550 1529.95
REMAINING LINE *CALL

OKIDATA
80 129.95
82 269.95
83 499.95
84 899.95
92 369.95
93 639.95
2350 1859.95
2410 2099.95

MANNESMAN TALLY
160L 539.95
180L 739.95
SPIRIT 80 269.95

OLIVETTI
P-41 *CALL

PANASONIC
KXP 1090 249.95
KXP 1160 999.95

QUADRAM
QUADJET 599.95

SANYO
5500 589.95

SHARP
125, 150 *CALL

SILVER REED
500 369.95
550 499.95
770 799.95

SMITH CORONA
TP-1, TP-2 *CALL

STAR MICRONICS
GEMINI 10 *CALL
GEMINI 15X *CALL
DELTA 10 *CALL

TOSHIBA
1350 *CALL

TRANSTAR
ENTIRE LINE *CALL

TERMINALS

ADDS
VPT-A1 449.95
VPT-A2 449.95
VPT-COL. 1049.95
VPT-3A 449.95

HAZELTINE
ESPRIT I 439.95
ESPRIT II 459.95
ESPRIT III 529.95
ESPRIT COLOR 759.95
ESPRIT 6310 459.95

LEAR SK-GLER
ADM-3A 509.95
REMAINING LINE *CALL

SANYO
COLOR RGB 519.95

TELEVIDEO
910 529.95
925 669.95
950 879.95
970 959.95

MULTI FUCTION BOARDS

AST
FULL LINE *CALL

QUADRAM
FULL LINE *CALL

GRAPHIC CARDS

AMDEK *CALL
HERCULES *CALL
PLANTRONICS *CALL
QUADRAM *CALL
USI *CALL

DISKETTES

FULL LINE *CALL

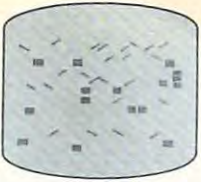
*CALL

WE DO NOT ONLY MEET PRICES... WE BEAT EVERY LEGITIMATE PRICE

800-346-5001
212-438-4111

TO ORDER-Simply dial toll free 800-346-5001 or 212-438-4111. You may order with Master Card or Visa, or you may send a money order, cashier's check, or certified chect to: **BSD ENTERPRISES**, 1781 64th St., Brooklyn, N.Y. 11204, and add approximate shipping, handling and insurance charges. Credit cards for phone orders only. All items subject to availability and price change. **NO TAX** for orders shipped out of state. **ALL MERCHANDISE SHIPPED BRAND NEW, FACTORY FRESH AND 100% GUARANTEED.** For customer service please dial **212-438-4111, ext. #17.**

Advertisers Index


Save UP TO 40% on COMPUTE!

Every issue of COMPUTE! contains up to 30 new programs and games. And a year's subscription brings them to you for less than 15 cents each! Plus you'll enjoy the most useful home computer advice, ideas and information anywhere! Subscribe now at up to 40% off the newsstand price. At less than 15 cents per program, this COMPUTE! offer is too good to pass up!

☐ 1 year $24—Save 32%! ☐ 2 years $45—Save 36%!
☐ 3 years $65—Save 40%!

Name ______________________

Address ______________________

City ____________ **State** ________ **Zip** ________

☐ Payment enclosed ☐ Bill me

Charge my ☐ Visa ☐ MasterCard ☐ American Express

______________________ ____/____
Account No. **Exp. date**

COMPUTE! brings you programs and games for the following machines: Atari, PET/CBM, VIC-20, TI 99/4A, Apple, Commodore 64, Radio Shack Color Computer, IBM PC and IBM PCjr.

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY CARD

FIRST CLASS PERMIT NO. 2312 GREENSBORO, NC

POSTAGE WILL BE PAID BY ADDRESSEE

COMPUTE!
P.O. Box 914
Farmingdale, NY 11737

The Editor's Feedback:

Computer: ☐ PET ☐ Apple ☐ Atari ☐ VIC-20 ☐ Commodore 64
☐ TI-99/4A ☐ Timex/Sinclair ☐ Radio Shack Color Computer
☐ Other ________________ ☐ Don't yet have one ...

Are you a COMPUTE! Subscriber? ☐ Yes ☐ No I would like to see:

More	Just Right	Fewer	
☐	☐	☐	Specific applications programs.
☐	☐	☐	BASIC programs.
☐	☐	☐	Machine language programs.
☐	☐	☐	Tutorials.
☐	☐	☐	Educational articles.
☐	☐	☐	Detailed explanations of programs.

More	Just Right	Fewer	
☐	☐	☐	Games.
☐	☐	☐	Reviews of game software.
☐	☐	☐	Reviews of business software.
☐	☐	☐	Reviews of educational software.
☐	☐	☐	Reviews of hardware.

What do you like best about COMPUTE! ?

What do you like least?

7 8 9 10 11 12 C

Place
Postage
Here

COMPUTE! Magazine
P.O. Box 5406
Greensboro, NC 27435-0406

COMPUTE!'s FREE Reader Information Service

Use these cards to request FREE information about the products advertised in this issue. Clearly print or type your full name and address. Only one card should be used per person. Circle the numbers that correspond to the key number appearing in the advertisers index.

Send in the card and the advertisers will receive your inquiry. Although every effort is made to insure that only advertisers wishing to provide product information have reader service numbers, COMPUTE! cannot be responsible if advertisers do not provide literature to readers.

Please use these cards *only* for subscribing or for requesting product information. Editorial and customer service inquiries should be addressed to: COMPUTE!, P.O. Box 5406, Greensboro, NC 27403. Check the expiration date on the card to insure proper handling.

Use these cards and this address only for COMPUTE!'s Reader Information Service. Do not send with payment in any form.

COMPUTE!

101	102	103	104	105	106	107	108	109	110	111
112	113	114	115	116	117	118	119	120	121	122
123	124	125	126	127	128	129	130	131	132	133
134	135	136	137	138	139	140	141	142	143	144
145	146	147	148	149	150	151	152	153	154	155
156	157	158	159	160	161	162	163	164	165	166
167	168	169	170	171	172	173	174	175	176	177
178	179	180	181	182	183	184	185	186	187	188
189	190	191	192	193	194	195	196	197	198	199
200	201	202	203	204	205	206	207	208	209	210
211	212	213	214	215	216	217	218	219	220	221
222	223	224	225	226	227	228	229	230	231	232
233	234	235	236	237	238	239	240	241	242	243
244	245	246	247	248	249	250	251	252	253	254
255	256	257	258	259	260	261	262	263	264	265
266	267	268	269	270	271	272	273	274	275	276
277	278	279	280	281	282	283	284	285	286	287
288	289	290	291	292	293	294	295	296	297	298
299	300	301	302	303	304	305	306	307	308	309
310	311	312	313	314	315	316	317	318	319	320
321	322	323	324	325	326	327	328	329	330	331
332	333	334	335	336	337	338	339	340	341	342
343	344	345	346	347	348	349	350			

Circle 101 for a one year new U.S. subscription to COMPUTE!: you will be billed for $24.

Please print or type your full name and address. Limit one card per person.

Name ____________________

Address ____________________

City ____________________

State/Province __________ Zip __________

Country ____________________

Please include zip code. Expiration 9/30/84 C0784

Place
Postage
Here

COMPUTE! Reader Service
P.O. Box 2141
Radnor, PA 19089

"The Complete CARDCO Line"

. . . and still growing!

CARDCO provides "Commodore-ready" computer accessories that will enhance your utilization of Commodore-64 and VIC-20 Computers, increase their capability, and add to your enjoyment and skill. AND, they're available for use with other personal computers, too.

Designed with the user in mind, CARDCO offers fine accessories including Printer Interfaces with and without graphics, Expansion Interfaces, Memory Expansions, Cassette Interfaces, Numeric Keypads PLUS "NOW" Software for your VIC-20 and C-64. These programs include the "WRITE NOW" Word Processor, "MAIL NOW" Mailing List, PRINTER UTILITY PROGRAMS on Tape and on Disk, "SPELL NOW" Spell Checker, "GRAPH NOW" including "PAINT NOW", and "FILE NOW".

CARDCO has three new Letter Quality PRINTERS with your choice of drumhead design (8 1/2" carriage), Daisy Wheel Design (13 inch carriage) and Daisy Wheel Design (11 inch carriage). "Commodore-ready" . . . plus; with compatible input for PC, PC jr., TRS-80 and many more personal computers. CARDCO's NEW "DATA CASSETTE RECORDER/PLAYER" is also "Commodore-ready" and ready for instant shipment at prices that will amaze you.

CARDCO will constantly increase its line with unique and new products to enhance the enjoyment of computer owners.

Write for illustrated literature and prices or see CARDCO Computer Accessories and Software wherever Computers are sold.

cardco, inc.

300 S. Topeka Wichita, Kansas 67202 (316) 267-6525

"The world's largest manufacturer of Commodore accessories."

Commodore 64 Magic Desk I

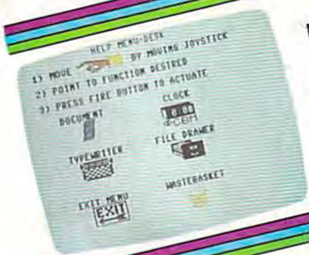
Only From Commodore—The Excitement and Simplicity of Magic Desk!

Only Commodore brings you the magic of MAGIC DESK . . . the next generation of "user friendly" software! Imagine using your computer to type, file and edit personal letters and papers *without learning any special commands!* All MAGIC DESK commands are PICTURES. Just move the animated hand to the picture of the feature you want to use (like the TYPEWRITER) and you're ready to go.

The MAGIC DESK Typewriter works just like a real ELECTRIC TYPEWRITER . . . and it's COMPUTERIZED. All the filing is *electronic*. Excellent sound effects and screen animation make typing fun, whether you're typing letters, reports or memos . . . and the built-in filing feature makes MAGIC DESK useful for keeping names and addresses, home inventory lists, insurance information and more.

Your COMMODORE 64, COMMODORE DISK DRIVE and MAGIC DESK are an unbeatable combination. Filing operations are automatically linked to your Commodore disk drive—but you don't have to know any commands—just "file" the pages you type in the file cabinet and your text is automatically saved on diskette. There are 3 file drawers with 10 file folders in each drawer and 10 pages in each folder.

To PRINT a page you've typed, just "point" at the picture of the printer and your pages are automatically printed on your COMMODORE PRINTER or PRINTER/PLOTTER. If you want to erase what you've typed, the WASTEBASKET under the desk lets you "throw away" pages. There's even a DIGITAL CLOCK which helps you keep track of time while you're typing.



Not only is MAGIC DESK easy to use . . . it's hard to make a mistake! Just press the COMMODORE key and one of several "help menus" appears to tell you exactly what to do next. Special messages show you how the various picture commands work and help you when you make a mistake. Help messages also show you how to use the printer, filing cabinet, digital clock and wastebasket.

Another reason why Commodore is number one in the world of microcomputers—Come join us.

commodore COMPUTERS

First In Quality Software