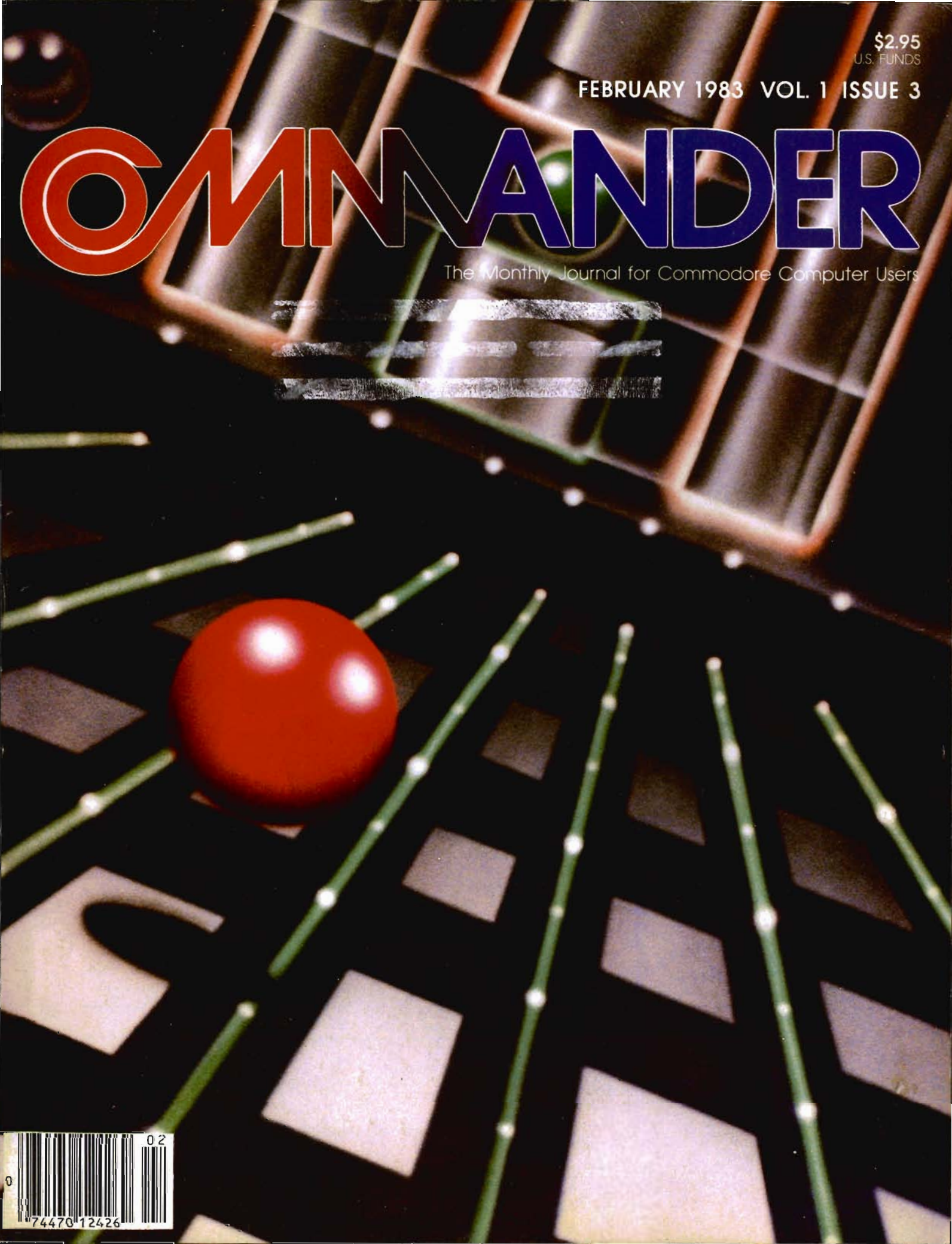

$2.95
U.S. FUNDS
FEBRUARY 1983 VOL. 1 ISSUE 3
COMMANDER
The Monthly Journal for Commodore Computer Users
0 74470 12426 02

UMI software...a world of choices

A World of Fun! They're hot! They're new! The exceptional graphics and challenging play of UMI's games have made United Microware the leader in arcade-quality recreational software.

A World of Help! UMI has created programs to help professionals and homeowners "take care of business." UMI can make your life a little easier with word processing, information storage, financial management, hobbyist programs, utilities and communication programs — all with easy-to-understand instructions.

A World of Choices! All programs come on cassettes or UMI's own durable cartridges, depending on your selection. If you're looking for fun, or for an easier way to manage your personal business, look to UMI . . . the leader you can trust. UMI products are available at your favorite computer products store.

Dealer inquiries invited.

umi

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

Table of Contents

STAFF

Publisher
THOMAS L. ROSENBAUM

Editor
ALICIA A. LINDEN

Dealer Manager
DEBBIE GABBARD

Sales Manager
TERILYN M. FLOYD

Advertising/Article Manager
LINDA L. LINDEN

Typesetting
GRANGE PRINTING

Consultant
EDWIN SUND
GEORGE R. GAUKEL

Printed By
GRANGE PRINTING

COMMANDER is published monthly by:
MICRO SYSTEMS SPECIALTIES, P.O. Box 98827,
Tacoma, Washington 98498

Subscription Rates	*Per Year*
U.S.	$22.00
Canadian, Mexican	$26.00
Surface Rates	$37.00
Air Mail	$54.00

For back issues, subscriptions, change of address or other information, write to:
COMMANDER
P.O. Box 98827
Tacoma, Washington 98498
(206) 565-6816

Call for Clubs and Newsletters Directory

To be included in the first edition of the Commander Clubs and Newsletters Directory, your club or publication must supply the following information:

1. *name of organization or publication*
2. *mailing address*
3. *contact person and telephone number*
4. *name of newsletter or publication*
5. *special interests*

Send your information to Clubs and Newsletters Directory, Commander, P.O. Box 98827, Tacoma, Washington 98498.

Commander—The Monthly Journal for Commodore Computer Users is published monthly by Micro Systems Specialties, P.O. Box 98143, Tacoma, WA 98498. Domestic Subscriptions, 12 issues, $22.00. Second Class Postage pending at Tacoma, WA 98404 and additional mailing offices. Postmaster: Send address changes to: **Commander—The Monthly Journal for Commodore Computer Users, P.O. Box 98827, Tacoma, WA 98498.**

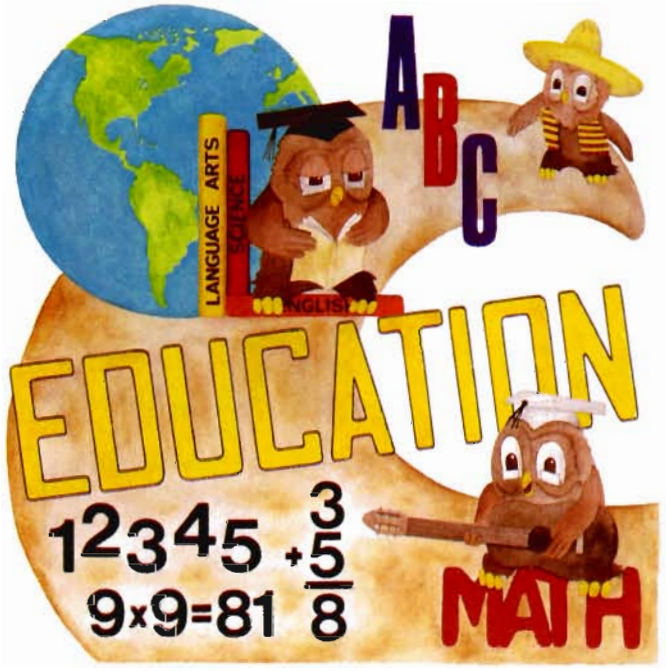
COMM*DATA SOFTWARE

GUIDES YOU AND YOUR VIC 20® DOWN ROADS OF ADVENTURE WITH:

- Maelstrom*
- Escape MCP*
- Gator Chase*
- Astro Command
- Caves of Annod
- Capture the Beast
- Whirlwind Rescue*
- Street Maze
- The Market
- Chivalry

THROUGH TRAILS OF CREATIVITY WITH:

- Sketch and Paint



ALONG THE PATH TO KNOWLEDGE WITH:

- Wordspot
- Math Tutor Series
- Alphabet Tutor
- Conversion
- Gotcha Math
- English Invaders
- Math Invaders Series

ASK FOR COMM*DATA COMPUTER HOUSE SOFTWARE AT YOUR LOCAL DEALER.

Or Send for FREE Catalog:

COMM*DATA COMPUTER HOUSE
320 Summit Avenue
Milford, Michigan 48042
(313) 685-0113

Dealer Inquiries Welcome.

Quality software also available
for Pet and Commodore 64 computers

MasterCard

VIC 20 is a Registered Trademark of Commodore Business Machines, Inc.
*High Res Full Machine Code Arcade Style Games.

VISA

Letters to the Editor

For you dear readers that have not yet figured it out, let me begin by saying that I am a very opinionated fellow and will be happy to express by opinion at the drop of a hat, or anything else covenient. Lately I have formed the opinion that Commodore could have done a slightly better job with our VIC had they only included a RESET button.

A RESET button is a very handy little device that could have been added in the manufacturing process for a few pennies or maybe a couple of dollars at most, but costs many dollars to have added later if one is not able to do it himself. All of the necessary parts can be obtained at *Radio Shack* for those that are able to do things like this and it is well worth the time. If you can't then I'd recommend taking your VIC to a **competent** technician and having a reset switch put in.

What is a reset switch and what is it used for? I'm glad you asked. A reset switch performs the same function as the on/off switch only it doesn't cause wear and tear on the sensitive power supply nor does it subject the sensitive CMOS circuitry inside the VIC to the transient voltages associated with turning the power off and on.

The reset switch performs its magic by applying a "ground" or low voltage, or "zero," to a line inside the VIC that is designed for that purpose. When you RESET your VIC, the screen shows the familiar:

```
****** CBM BASIC V2 ******
3583 BYTES FREE
READY
```

6

just as if you had turned it off and on.

Many of you may not see the advantage to the reset even now. The only way I can explain is to remind you of the times that you have had a game loaded in that wouldn't repond to the RUN/STOP and RESTORE keys and the only way you had to get control of your VIC back was to turn it off. Another, perhaps less obvious, advantage is for those of you that have "mother boards" and want to switch games or cartridges in and out. Several mother boards come with reset switches on them but if you have one that doesn't (as I do) then you can now have a RESET to play with just like the big boys. The way to accomplish this miracle—simplicity itself as Commodore brought the necessary lines right out to the expansion port so all that needs doing is to follow these simple steps;

1. Carefully take your VIC apart.

2. Locate a convenient mounting spot for momentary switch.

3. Locate pins X and Z on the bottom of the board at memory expansion port.

4. Run one wire from pin X to one side of your switch, run another wire from pin Z to the other side of your switch.

5. Carefully reassemble VIC.

That should take care of that. There is an alternate method, and that is to mount your switch on your mother board if you have one. Additionally, you can always take your VIC to the **competent** technician mentioned above.

On another subject entirely, for those of you that may have read any of my previous OPINIONS you know that I have had a very difficult time obtaining a 16K memory expansion because the company I originally ordered from didn't have it in stock, etc., etc. Well, I finally got a 16K expansion and was able, at last, to run my 'VIKING!' game that I had obtained from Prickley-Pear Software. I want you to know it was worth the wait.

'VIKING!' is an adventure type game that is **available** on disk, requires 16K of memory expansion and consumes HOURS. It can be played by one to four players and the program asks if each player is "man or woman" and then proceeds through the rest of the game with that information.

The play is engrossing, everyone starts out as equals in the year 750 in Norway as "holders" with a certain amount of land, a boat, a few people and a few other items. The object of the game is to acquire land, people and goods in sufficient quantity to be promoted, ultimately, to King and thus win the game. It takes a very careful and skillful player to get the right items at the right time without some disaster befalling you such as plague, raiders, tax revolt, bankruptcy, poor harvests or other pitfalls. All in all a thoroughly enjoyable game and one I would recommend.

We're also tried out the Discwasher "Pointmaster" joystick, available from Discwasher. The jury is still out around here on this one, about all I can say for it right now is that the feel is "different." The kids seem to like it though and they are the ones that will use it the most so maybe my opinion will improve as I become more familiar with it.

Until we meet again, that's my opinion, what's yours???

by Fred S. Dart
Salem, Utah

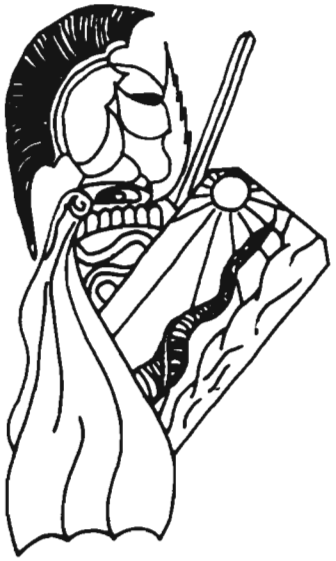
Editorial

AN APOLOGY TO JIM STRASMA

It has come to our attention that Mr. Jim Strasma, editor of the Midnite Software Gazette, has been unable to contact us by mail. I sincerely regret this occurence and hope that it will not happen again, but I have no record of having received any letter from Mr. Strasma. I have no intention of blaming the U.S. Post Office for this fiasco and accept full blame myself. Our lifeblood is the Commodore public and Commander will NEVER knowingly ignore the well intentioned request of any person, least of all someone of Mr. Strasma's stature. We strive to serve our readers to the very best of our ability and will reply to every letter which we receive.

I apologize for this mixup, Mr. Strasma, and hope to be of service to you in the future.

—Publisher



We at Commander are very excited about some new departments which we are beginning this month. We received an enthusiastic response to our call for information on Commodore Users' clubs and want to provide a forum for the public exchange of information on other subjects in order to foster a spirit of cooperation among Commodore users worldwide. There are many people in various parts of the world with unanswered questions about their computer systems and just as many people who know the answers to their questions. We intend to bring the askers and the askees together in an effort to make computing more fun and enjoyable and less frustrating.

Our new Bits and Pieces Department will spotlight questions put forth by our readers and feature the answers provided by other readers or one of Commander's technical experts, if necessary.

In order to make this project a success, we will need the help of our readership. No question should be considered too large or too small. Technical, non-technical, educational, business and recreational questions are all fair game and should be submitted. A good example would be information exchange between the scattered Superpet owners in the States and their Canadian compatriots. There is generally more Superpet knowledge in Canada since the Superpet was designed and developed there.

The Live Wire Department will focus on the popular field of electronic bulletin boards—if you or your organization are running an electronic bulletin board, let us know the particulars of your system so that we can spread the word.

There is still no solid word from Commodore on whether or not they will market the Max Machine in the U.S. We have seen a recent news-release for the Max in an American magazine which leaves one with the impression that you should be able to buy one. Commodore public relations officials state that the release of the Max has been postponed indefinitely, which is not surprising in light of the fact that the VIC-20 can be purchased at discount stores for less than $160.00. The Max is selling well in Japan at this time and we hope that Commodore will release it in the U.S. soon.

One last hot tidbit this time around—Commodore and Zilog has reached an agreement and we should see a Z8000 based, low cost 16 bit microcomputer from Commodore in the future. This deal was a fine example of a true symbiotic relationship—Commodore did not want to spend the time and/or money to develop a 16 bit microprocessor and Zilog desperately needed a major OEM to go for the Z8000 before Motorola's 68000 completely buried it. The first inexpensive, powerful, 16 bit personal micro will definitely carve a niche for itself in the marketplace—look for Commodore to release this baby in the third quarter coupled with a price drop on the 64 in order to establish a market position for each machine.

New Products

CX6401B COMMODORE 64/VIC-20 PARALLEL PRINTER INTERFACE

ECX COMPUTER COMPANY announces its new Model CX-6401B printer interface for the Commodore 64 and VIC-20 computers. The CX6401B interface allows these computers to use ANY type of standard commercially available parallel interface type printer. This new interface is fully compatible with the popular WORDPRO 3+ wordprocessing program. In addition CX6401B also has all of the features of the original CX6401 model.

The CX6401B interface connects to the SERIAL PORT/BUS on the computer, NOT the user port, allowing the user port to remain open for use by a MODEM or other RS-232C peripheral device.

Some of the key features of the CX-6401B are:

* Compatible with WORDPRO 3+ wordprocessing program.
* Eliminates hand shaking problem of RS-232C cartridge.
* Mode switch to correct ASCII problem.
* No expensive RS-232C cable necessary.
* Lower cost than RS-232C type hook-up.
* Allows use of lower cost parallel type printers.
* No external power required.
* Allows use of multiple printers.
* Unused user port lines available for external usage.

The CX-6410B comes standard with a 3 ft. serial bus cable and connector for direct connection to the Commodore 64 or VIC-20 and a 1 ft. parallel cable for connection to the printer.

Suggested Retail Price: $89.95.

For more information contact:

ECX Computer Company, 2678 North Main St., Walnut Creek, Ca. 94526. (415) 944-9277.

CX6405 COMMODORE 64/VIC-20 EPSON MX SERIES PRINTER INTERFACE

ECX COMPUTER COMPANY announces its new Model CX6405 EPSON printer interface board. The CX6405 allows the new Commodore 64 and VIC-20 computers to connect directly to any model of the EPSON MX series of printers, including the new high speed FX-80 F/T. The CX6405 installs internally within the EPSON MX series printers. This new interface is fully compatible with the popular WORDPRO 3+ wordprocessing program.

The CX6405 interface connects to the SERIAL PORT/BUS on the computer, NOT the user port, allowing the user port to remain open for use by a MODEM or other RS-232C peripheral device.

Some of the key features of the CX6405 are:

* Compatible with WORDPRO 3+ wordprocessing program.
* Eliminates hand shaking problem of RS-232C cartridge.
* Mode switch to correct ASCII problem.
* No expensive RS-232C cable necessary.
* Lower cost than RS-232C type hook-up.
* No external power required.
* Allows use of multiple printers.
* Unused user port lines available for external usage.

The CX6405 comes standard with a 3 ft. serial bus cable and connector for direct connection to the Commodore 64 or VIC-20.

Suggested Retail Price: $79.95.

For more information, contact:

ECX Computer Company, 2678 North Main St., Walnut Creek, Ca. 94526. (415) 944-9277.

The AVALON HILL Game Company

Perhaps it is pure luck or even sheer genius but EIGHT MICROCOMPUTER GAMES CASSETTES FOR THE PET 2001 are ALSO COMPATIBLE WITH THE HOT, NEW COMMODORE 64 COMPUTER!

The following multi-cassette games are ready to run on the "64":

#40001 B-1 Nuclear Bomber $16.00
#40101 Midway Campaign .. 16.00
#40201 North Atlantic Convoy Raider 16.00
#40301 Nukewar 16.00
#40401 Planet Miners 16.00
#41601 Draw Poker 16.00
#40901 Computer Stocks & Bonds 20.00
#42001 Andromeda Conquest 18.00
#41501 Computer Football Strategy 16.00
#42401 Telengard 24.00

We share Commodore's praise of their new "64" as "The machine their competitors couldn't do at half the price." And Avalon Hill is ready and able to supply your software needs with strategy games that already have proven sales performance.

New Product Information

Please note that Cyberia Inc., Ames, Iowa, has released a farm accounting package for the Commodore 64 computer. Cyber-Farmer 64 is a new ver-

sion of the Cyber-Farmer program which has been in use by many farmers in the midwest for more than two years. The retail price of Cyber-Farmer is $195. It is available through Commodore dealers or directly from Cyberia Inc.

Sprite Shaper™

The Sprite Shaper™ program allows the Commodore 64 owner to quickly and easily design sprites (movable object blocks) for use in game programs or other applications. No graph paper is needed. No math is necessary to use the program.

The Sprite Shaper has many useful features, yet it is very simple to use. Many safeguards and provisions have been built into the program to protect against user errors and to increase program speed.

One of the unique features of the program is the constant display of the sprite being edited in all four possible sizes. Any change to the sprite will be immediately shown in all four sizes of sprites. This is a great help in designing sprites. The sprite color and background color can quickly be changed to see how different color combinations will appear. Up to four different shapes can be designed and saved to either printer, (VIC-1525) disk drive, or cassette. With the Deluxe Version, shapes can be reloaded later for further editing.

A menu allows you to choose color, shape number, background or sprite priority, and size. The program automatically calculates the poke statements for your program. No more binary arithmetic necessary for most applications!

Each Sprite Shaper Program comes with a thorough owner's manual that gives a complete lesson on using sprites as well as tips on using the program.

The deluxe version allows the use of three colors per sprite. Each individual sprite color can be changed to try different effects. The constant display of the four sizes of sprites is very useful in designing multi-color sprites.

Regular Version

Cassette $24.95
Diskette $26.95

Deluxe Version

Cassette $29.95
Diskette $31.95

Shipping to USA and Canada included in above prices.

Sound Shaper™

The Sound Shaper program allows you to fully explore the capabilities of the 6581 Sound Interface Device (SID chip). This chip has all the features and capabilities of a music synthesizer. The Sound Shaper program allows you to adjust various features of the SID chip quickly and easily.

Four different waveforms can be chosen. Different attack, decay, and release rates, as well as different sustain levels, can quickly be chosen. Dif-

Continued on page 67

News Releases

Superscript

The Management Accountability Group, Inc. (MAG) has acquired marketing rights to Precision Software's new word processing software "Superscript." Under two separate agreements MAG will: 1) distribute the Commodore Business Machines version of Superscript throughout six southern states (North Carolina, South Carolina, Georgia, Florida, Alabama, and Tennessee) and 2) convert the program to Texas Instruments Business Systems and market it nationally.

According to Peter Masterman, Vice President of Operations, word processing marketers have traditionally looked at themselves as "God's gift to computer retailers." Word processing has been sold like a commodity, thus carrying very low margins for the computer dealer. MAG plans on changing that by offering their Superscript dealers significant margins. This will enable dealers to provide better service to their end users.

After several reviews of Superscript, MAG is convinced it is as powerful as any word processor on the market today including Word Pro 4 Plus, Magic Wand, and Word Star.

Secure

A new program encryption kit manufactured in England by Computer Applied Technology is now available in the United States. Called SECURE, the encryption kit is endorsed by Commodore and produces 256 encryptions of single programs at random. Before decrypting programs, SECURE performs a series of checks to insure that the system is normal and then continually monitors for improper hardware or software interruptions. SECURE will not react to any device or any routine which is call enabled from a protected program. The unit is compatible with BASIC, Composite and Machine Codes, and consists of a leather-walleted key, a cassette or disk, a comprehensive manual and registration card.

Priced at $100.00 retail, SECURE is available from a San Diego-based firm called Distribution Unltd., P.O. Box 81702, San Diego, CA 92138-1702. (619) 299-3718. Dealer inquiries welcome.

National Educational Computer Library

NECOL is a non-profit educational organization that serves as a national resource center, assisting schools in their educational computer goals. Among other activities we review software and computer books to help schools locate suitable educational computer programs and we publish bibliographies on articles written in magazines relating to computers in education. Basically we act as a national specialized informational center assisting schools in filling their educational computer needs. In conjunction with the above we publish the National Educational Computer Review, a tabloid newspaper which is distributed free of charge to over 30,000 public and private grammar schools and high schools nationwide. National Educational Computer Library, 16 Main Street, New Milford, CT 06776. (203) 354-7760.

Software in The United Kingdom and Middle East

The Management Accountability Group, Inc. (MAG) has consumated agreements to market their small business software in the United Kingdom and the Middle East. The programs will be available for the IBM PC, Texas Instruments 300, 600, and 800 Series, Victor 9000 and Commodore Business Machines computer lines.

The MAG software will be distributed on a royalty basis allowing MAG to maintain control over marketing efforts.

According to Peter Masterman, Vice President of Operations, MAG has declined several offers to purchase foreign marketing rights to MAG software. "MAG is and will remain a marketing organization. We began developing business software because of the lack of good quality, fully integrated business software for micro computers."

For further information, please contact Peter H. Masterman, Vice President of Operations.

TYPE-SHARE TYPESETTING
EDUCATIONAL SPECIALISTS
SILICON OFFICE SPECIALISTS

SOUTHERN CALIFORNIA
MAINTENANCE CENTER
FOR COMMODORE EQUIPMENT

ALL UNDER ONE ROOF

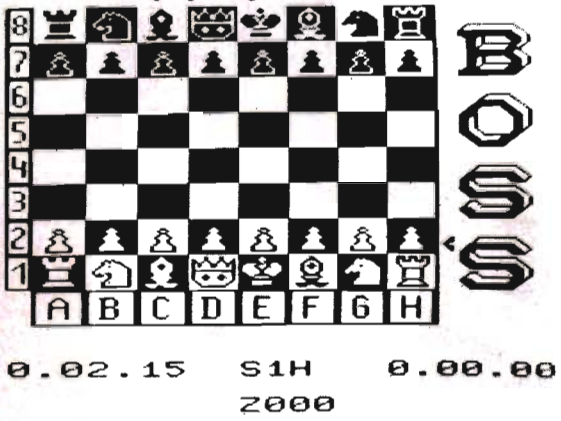
ONE STOP CENTER
for
commodore

VIC-VILLE ™ SOFTWARE

division of Data Equipment Supply Corporation

BOSS (c) by Kavan Software

Exclusive distributors of Kavan Software



The Definitive Chess Game for the VIC-20

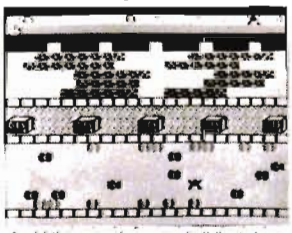
- ★ 10 Levels of Play
- ☆ Beats Sargon II
- ★ Two Clocks
- ☆ Wide range of opening moves
- ★ En passant, queening, castling
- ☆ Change screen and board colors
- ★ Cassette
- ☆ Requires 8K minimum expansion
- ★ 100% machine language

$39.95

BONZO (c) by Kavan

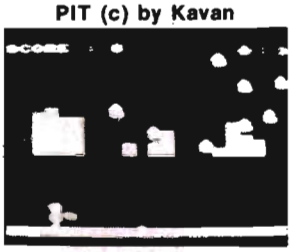
One of the most popular games in Europe. You control BONZO as he climbs the ladders and picks up the point blocks. Watch out for the alien guards. 100% machine language, cassette based. Joystick or keyboard, minimum 8k expansion. **$20.00**

HOPPER



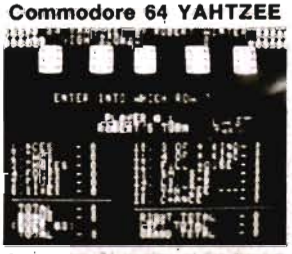
Avoid the cars, dragsters, buildings, logs and other obstacles to bring the frog safely home. Machine language for fast and smooth arcade action. Joystick, standard VIC. **$20.00**

PIT (c) by Kavan



BONZO strikes again as he takes money bags out of the pit. Avoid the alien rain by standing under the shields. Every successfully removed bag of money reinforces your sheilds. 100% machine language, cassette based. Joystick or keyboard, standard VIC. **$18.00**

Commodore 64 YAHTZEE



Commodore 64 version of the famous dice game. 10 player capacity. Watch dice roll across the screen. Automatic tabulation of score and bonuses. Sprite graphics and sound. Cassette based. **$20.00**

Night Crawler **$25.00**
by Interesting Software
Shoot down centipedes, spiders, mushrooms and all kinds of bugs before they get you. Machine language arcade action on standard VIC with joystick.

The Black Castle **$20.00**
Adventure, travel the countryside, fight demons, buy goods, storm the castle. Requires 3k or more expansion.

A Maze Ing **$12.00**
Travel through the maze. Game of skill and tense action. Standard VIC.

Gobbler **$11.00**
Sounds easy? You have 25 seconds to get him and the time gets shorter at each higher level. Standard VIC.

Hang U **$12.00**
Traditional Hangman plays against the VIC's 250 word dictionary or another person. Standard VIC.

Coggle **$11.00**
Computerized version of Boggle. Standard VIC.

Gold Brick **$14.00**
Many levels of play, sound, and color.

3-D Labyrinth **$14.00**
Escape from the labyrinth. Shown in 3-D perspective view with randomly generated mazes. Standard VIC.

Air Strike **$11.00**
Fly the new super bomber V-20 on a mission. Standard VIC.

Attack on Silo III **$12.00**
You are the commander of Silo III. Defend your country. Standard VIC.

Baseball Strategy **$12.00**
The excitement of baseball as a video strategic game. Standard Vic.

Vic Poker **$14.00**
Play poker against the VIC. Hi-res graphics and sound. Standard VIC.

Frogger by (c) Kavan **$14.00**
Eat the flies and avoid the car. Standard VIC.

Space Phreeks **$25.00**
by Interesting Software
Pilot the spaceship "Infinity" and fight the "Space Phreeks". 15 different attack patterns, 33 levels. Machine language, arcade quality. Standard VIC, joystick.

Mailing List **$25.00**
Keep mailing list, print reports, labels. 8k expansion or 16k expansion required.

Astro-Miners **$17.00**
Hi-res graphics and sound space game. Requires 3k or 8k expansion.

Panzer Attack **$14.00**
Enemy tanks are attacking and you must destroy them. Hi-res graphics. Standard VIC.

Pedestrian Polo **$14.00**
Drive your car thru the streets. Based upon Death Race. Standard VIC.

Yahtzee **$12.00**
Solitaire version of this famous dice game. Standard VIC.

Commodore 64 Software **Available Now !!**
64 Monopoly from AP Software
64 Mailing List from VIC-VILLE™ Software
64 Finance from VIC-VILLE™ Software
64 Time Manager 2.0 from TOTL Software

Look for more 64 Software from VIC-VILLE™ & get on our mailing list for all 64 updates and users' group.

Complete descriptive catalog $2.00

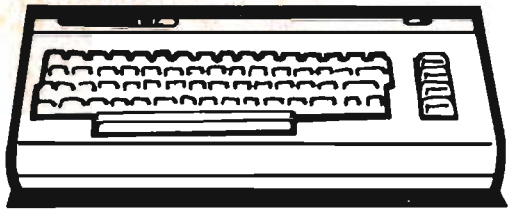
Dealers Welcome - Authors Wanted !

Add $3.00 for shipping & handling

Network your CBM, VIC and COMMODORE 64 with the PET SWITCH and VIC SWITCH from DATATRONICS.

Distributors for Datatronic AB

(714) 778-5455 — DES — Data Equipment Supply Corp. 8315 Firestone Blvd., Downey, CA 90241 — (213) 923-9361



VIC 20

DOSPLUS: A DOS Support Utility for the VIC-20

by Eric Giguere
Alberta, Canada

For those of us lucky VIC owners who have a disk drive, it has to be admitted that it's probably one of the best things we did for our computer (and ourselves!). The increased speed and storage capacity sure make a difference over cassettes. But as for ease of use, well, it has to be admitted that it could have been done better. The manual is very hard to follow and instructions not explained well or in detail. And for those of use who have worked on Apples and TRS-80s, the way that you control the disk is totally unfamiliar. It takes quite a bit of work just to prepare a disk for loading or saving. That is why I created DOSPLUS—a DOS Support program for use with the first disk drive (device no. 8). I'm sure that you'll find it very useful, because I have. So if you're interested, read on!

Why a DOS Support Program?

I was sitting at my computer one Sunday afternnon, with nothing too important to do, when I decided to created a DOS support system. I had gotten sick and tired of having to OPEN a file each time I wanted to send a command to the disk drive, and so decided to do something about it. Thus, in the time from 2 p.m. to 4 p.m., DOSPLUS was created!

For those of you who don't yet know the intricacies of Commodore's DOS (Disk Operating System), it can be described, as I mentioned above, as cumbersome. To communicate with the disk, you have to OPEN a file in BASIC and then send the command to the disk through the file via a PRINT# command. An example would go as follows:

```
OPEN15,8,15
READY.
PRINT#15,"INITIALIZE"
READY.
```

What happened here is that you OPENed a file to what is called the "command channel" of the disk drive. Whatever you send to this command channel will be interpreted as a command to be executed, not as data. Thus you are ready to send a command, as flagged by the READY given by the VIC. (For those of you who don't know what the three numbers following OPEN are, they signify the following: 15 = file no., a reference for both you and the VIC; 8 = device no., which in this case means the first disk drive; and 15 = command channel, which allows you to send commands to the disk. Notice that the file no. is the same as the command channel. This allows us to easily remember which file gives access to the command channel. The result of the OPEN command will then be to OPEN logical file number 15 which will access the command channel 15 of the device number 8, the disk drive.) What you must now do is send the command, and this is achieved by using PRINT#. In the example we PRINT#15,"INITIALIZE". What this does is send the command "INITIALIZE" through file number 15 to the disk drive. The DOS in the drive will then interpret the command and proceed to execute it. (INITIALIZE simply prepares the disk drive for the use of a different disk. It does **not** format a disk, as in other DOSs.) As you see, this is a very unwieldy way of telling the disk that a new floppy has been placed in it.

How to Use DOSPLUS

As opposed to constantly OPENing and PRINTing to a file, DOSPLUS allows you to hit one key and then type your command. When you hit the RETURN key, the command will automatically be sent to the disk drive and executed (as long as it is a legal command). This greatly simplifies the use of the disk drive, as any command can now be instantly sent with only a few keystrokes. Following is how to use DOSPLUS:

1) First activate DOSPLUS, using the instructions in the following part.

2) Next, hit the right bracket key (']'), by using SHIFT and the semicolon key.

3) Type in the DOS command, excluding the PRINT# and quotations. Example: PRINT#15,"INITIALIZE" would become]INITIALIZE.

4) After making sure everything is OK, hit the RETURN key. PRESTO! The disk drive comes to life and executes your command. Isn't that easy? And it works for every command, including NEW, COPY and SCRATCH. Now life is really easy. One note though: DOSPLUS does not work in a program. You can only use it in the direct mode, when not using line numbers. In a program you must use the procedure of OPENing and PRINTing to files as usual. But then, I didn't promise you perfection!

DOSPLUS also adds two new commands to your repertoire, though they really aren't commands. The first is the 'E' command. This stands for 'Error' and prints to the screen the present error status of the disk drive. So instead of having to type in

```
10 OPEN15,8,15:
INPUT#15,A,B$,C,D:
PRINTA;B$;C;D
```

and then running that line to get the

present error condition of the drive, you simply type:

]E (RETURN)

and the same information will be displayed on the screen. To demonstrate this, type in the following:

OPEN1,8,2,"XYZ" (RETURN)

Now, assuming that you don't have a file by that name on the disk, the error light on the drive will begin to flash, signaling some kind of error. You then have two options to find out what kind of error it is. Either type in line 10 above or simply use]E (RETURN). Both will give you

62, FILE NOT FOUND,00,00

or some variant, but which is the easier to use? Unless you like doing things the hard way, I'm sure you'll agree that the E command is the easiest to use.

The second "command" added by DOSPLUS is 'K', for Kill. It disables DOSPLUS, which means that you won't be able to use the right bracket key anymore to send disk commands. Why is it needed, you ask? The fact is, if you load a program from tape or use the cassette recorder in any way, you will destroy the machine language program and effectively crash the computer. This is because the computer jumps to the cassette buffer when DOSPLUS is activated, where it encounters some machine language code that makes it do abnormal things, like sending DOS commands to the disk drive. If you use the tape recorder in any way while DOSPLUS is activated, the cassette buffer is replaced with information from or going to the tape recorder, and so destroys your program. When the computer then jumps to the cassette buffer, it will only find garbage and may then crash. That is why it's important to be careful in the use of the tape recorder while DOSPLUS is active. The use of the 'K' command (format:]K) prevents all of this. (You may ask why I placed DOSPLUS in the cassette buffer. There are two reasons: firstly, if I had placed it at the top of memory, there would have been conflict with programmable characters and BASIC, and secondly, because you need a disk drive to use DOSPLUS, you probably won't use your tape recorder. Thus I felt that the cassette buffer was the ideal place to put DOSPLUS.)

How to Type In DOSPLUS

DOSPLUS is, as mentioned before, a machine language program. This means that it cannot be simply typed in as a BASIC program would. Instead, you can either use a monitor and do it using memory dumps, or you can poke the instructions into memory via a program. Program #2 is just that—a program that pokes DOSPLUS into memory. It also saves it to disk for you, which cannot be normally accomplished from BASIC, and so saves you a lot of mess. Simply type in DOSPLUS/DATA, RUN it, and you can save DOSPLUS to as many disks as you wish. I have tried to make the program as user-friendly as possible, and so make it easy to abort the program if you make a mistake. It also provides error messages in case anything goes wrong.

Program #1 can only be used once you have a disk with the machine language program saved on it, using DOSPLUS/DATA. Type it in, and then save it to disk, as it automatically NEWs itself when finished. This program is a BASIC loader program which loads the actual machine code (saved as DOSPLUS.OBJ by program #2) into the computer, activates it by a SYS828, and then resets the program pointers by executing a NEW, which also clears the program. It simplifies loading DOSPLUS.OBJ because all you have to do is LOAD the program (saved as DOSPLUS—but remember, it isn't the actual program, but a BASIC loader. The actual program goes by as DOSPLUS.OBJ) and run it. If everything went OK, the computer should respond with the message "DOS OK." This is your signal that DOSPLUS is up and running, and that you can now use the ']' key to send disk commands. That's all there is to it! Once you've loaded and activated the machine code, it becomes easy as pie to use all those great disk commands.

For those of you who like doing it the hard way, you can also load and activate DOSPLUS using the following:

```
LOAD"DOSPLUS.OBJ",8,1
READY.
NEW
READY.
SYS828
DOS OK.
READY.
```

This has the same effect as loading and running DOSPLUS.

Program details:

DOSPLUS—this is the BASIC loader for DOSPLUS.OBJ, and is only three lines long. Line 10 checks to see if DOSPLUS is already in memory, and if not, loads it. The line must be typed in exactly as is, otherwise it will not work, and the VIC will be constantly loading and re-loading the machine code. Line 15 pokes the keyboard buffer pointer with 6, and then pokes six numbers into the buffer itself, which starts at 631. What this does is, upon the end of the program, act as if you had typed in SYS828 in direct mode. If you examine the numbers being poked into the buffer, you'll notice they represent the ASCII of S, shifted-Y,8,2,8, and RETURN (the key, not RETURN after GOSUB). When the program encounters line 20, it erases itself and then checks location 198 to see if there are any numbers stored in the keyboard buffer, and if there are, print them on the screen as their ASCII representations. Since we poked a 6 in 198, it will print the first six keys it finds in the buffer, which we have made to be S(shift)Y828 and RETURN. Because of the RETURN at the end, the line is entered and executed by the computer, as if we had typed it in ourselves. The screen should then show "DOS OK.", indicating that DOSPLUS is ready to be used.

DOSPLUS/DATA—This is a more complicated program which both POKEs the machine code into memory and saves it to disk as many times as you want it to. Lines 10 to 30 read the data and poke it into memory, making sure that it is correct by the use of a checksum. If it encounters any errors, the program stops. Lines 40 to 65 display a prompt to save DOSPLUS to disk by hitting RETURN, or to abort the program using the F7 key. Lines 70 to 90 do the actual saving. First the program opens a program file on disk called 'DOSPLUS.OBJ'. Then it reads the error channel to make sure there aren't

any error conditions—if there are, the program aborts. The program then sends the starting address of the program to the file, which in this case equals 3*256 + 60, or 828, in standard low-byte, high-byte order. The data is then read and sent to the disk by line 80. Finally, the computer prints that the save is successful and closes the files. Line 95 asks if you want to save it again to another disk, and if not, ends with line 99. Lines 100 to 250 store the machine code as DATA statements. There are ten such statements on every line, plus a checksum, which is a total of all the preceding numbers on that line, and is used to check that the data has been entered correctly. A note of warning: if you don't type in the checksum correctly, the program could give you errors, or your data may be incorrectly saved to disk. Please type these in carefully.

The Program Itself:

This part is only meant for those who know something about assembly language, as it details how DOSPLUS.OBJ works. Feel free to skip this part if you wish.

Program #3 is the actual disassembly of DOSPLUS itself, as it resides in the cassette buffer. If you have the VICMON cartridge, you can type it in exactly as it is. I have written in some comments beside the actual code, so that you can see exactly how it works, though I'm going to give a brief explanation right now.

First, DOSPLUS places a JMP $0359 in the CHRGET routine which starts at $0073. This routine is responsible for getting the next character for the BASIC interpreter to handle, and because it is located in RAM, we can fool around with it. After the program has placed the JMP in the proper address, it loads Y with $03, and A with $4F, and then jumps to the routine at $CB1E, which prints the message it finds at the address specified by Y and A, which in this case equals $034F. Note that the bytes from $034F to $0358 don't represent any actual code but instead are the data for DOS OK. in ASCII, preceded and followed by $OD, ASCII for carriage return. The program then exits back to BASIC after having printed the message.

PROGRAM 1

```
1 POKE36879,27:PRINTCHR$(145)"■■DOSPLUS
SAVER UTILITY ————————————————————————"
5 PRINT"CHECKING MEMORY."
10 FORI=100TO259STEP10:B=0:FORJ=0TO9:REA
DA:POKE728+I+J,A:B=B+A:NEXTJ
20 READC:IFB<>CTHENPRINT"■ERROR IN LINE"
;I:STOP
30 NEXTI:PRINT"■OPERATION SUCCESSFUL."
40 PRINT"■PLEASE INSERT DEST.   DISK AND
 PRESS ■■■■■ (■■ TO ABORT) ==> ";
45 X$="■ ■":Z=0
50 T=TI+18:Z=1-Z:PRINTMID$(X$,Z+1);"■■";
55 GETA$:IFT<TITHEN50
60 IFA$=""THEN55
65 PRINT" ":IFA$="■"THENPRINT"■■■ABORT.■
":END
70 OPEN15,8,15,"I":INPUT#15,E:IFE>19THEN
PRINT"■DISK I/O ERROR.":CLOSE15:STOP
75 OPEN1,8,2,"@:DOSPLUS.OBJ,P,W":PRINT#1
,CHR$(60);CHR$(3);
80 RESTORE:FORI=0TO15:FORJ=0TO9:READA:PR
INT#1,CHR$(A);:NEXTJ:READA:NEXTI
90 PRINT"■SAVE SUCCESSFUL.":CLOSE1:CLOSE
15
95 INPUT"■TRY AGAIN";A$:IFLEFT$(A$,1)="Y
"THENGOTO40
99 END
100 DATA169,76,133,124,169,89,133,125,16
9,3,1190
110 DATA133,126,168,169,79,32,30,203,96,
13,1049
120 DATA68,79,83,32,79,75,46,13,0,201,67
6
130 DATA93,208,8,72,165,123,201,2,240,9,
1121
140 DATA104,201,58,144,1,96,76,128,0,104
,912
150 DATA32,115,0,144,19,201,75,208,20,16
9,983
160 DATA201,133,124,169,58,133,125,169,1
76,133,1421
170 DATA126,76,116,196,162,11,76,58,196,
133,1150
180 DATA0,169,0,32,189,255,169,127,162,8
,1111
190 DATA160,15,32,186,255,32,192,255,165
,0,1292
200 DATA201,69,240,28,162,127,32,201,255
,165,1480
210 DATA0,32,210,255,32,115,0,201,0,208,
1053
220 DATA246,32,204,255,169,127,32,195,25
5,76,1591
230 DATA116,196,162,127,32,198,255,169,1
3,32,1300
```

Following this and starting at $0359, comes the routine which is jumped to by CHRGET after we placed our 'wedge'. This routine checks for the right bracket key and if the computer is presently in direct mode. If not, it processes everything as per usual, and eventually returns from the routine. If the right bracket has been hit and the computer is in direct mode, then the routine jumps to $036D, which then processes the next character it finds. If 'K', then it kills DOSPLUS by replacing the JMP we put in CHRGET with the normal values. Not that it does **not** erase the program, and so can be reactivated using SYS828 from BASIC OR .G 033C from monitor.

The rest of the program is pretty well self-explanatory. It opens a logical file to the disk (no.127) command channel, and then opens it either for input or output, depending on whether the first letter after the right bracket is an 'E' for 'Error' or another letter to be sent to the disk. If the letter is E, it will input the error message from the disk, print it out to screen, and return to BASIC. Otherwise it will take what you have typed in and transfer it to disk as a command, and then return to BASIC. All this is done through the use of Kernal routines, as shown in the listing.

Note: For those of you who are interested, there are two useful routines accessed in the program. The first is the routine at $C474. A JMP to this location will return you to BASIC, print the 'READY.' message and turn on the cursor as if nothing had happened. The second routine is also useful. It prints out the error message corresponding to the number in the X register. Error number 11 ($10B) is ?SYNTAX ERROR, and this is used by the program when you type a number afrer the ']' key. Other messages can be found by loading the X register with a different value, and so can be used by any machine language routine that uses error checking.

A Final Note

DOSPLUS is meant as a DOS support program, and does exactly what it is supposed to. I gave it the name DOSPLUS because I have not heard of any program by that name. Feel free to call it anything you wish, so long as you make the appropriate changes to the programs. Why, you could call it DOSMINUS for all I care! Seriously, though, I have to admit that the program is far from perfect. For one thing, it doesn't make any syntax checks except to make sure no numbers are sent as the first characters to disk. If you type in "]STUPID", then that is exactly what DOSPLUS will send, and you'll get an error message. In any case, I feel that it is a practical program worth typing in, because it saves you a lot of trouble. But if you don't like typing in programs, then send me a **formatted** 1540/1541 disk, a self-addressed envelope or mailer (no stamps please), and $7 to: Eric Giguere, Box 901, Peace River, Alberta, Canada TOH 2X0, and I'll send you all three programs as soon as possible. (Please mark the envelope "Magnetic Materials—Do Not Fold.")

```
240 DATA210,255,32,207,255,32,210,255,20
1,13,1670
250 DATA208,246,240,223,0,0,0,0,0,0,917
```

PROGRAM 2

```
10 IFPEEK(828)<>169THENLOAD"DOSPLUS.OBJ",
8,1
15 POKE198,6:POKE631,83:POKE632,121:POKE6
33,56:POKE634,50:POKE635,56:POKE636,13
20 NEW
```

(Program 3 see page 64)

An Introduction to Assembly Language Programming on the VIC-20 Part III: Registers and Addressing Modes

by Eric Giguere
Alberta, Canada

In last month's column we took a look at monitors and assemblers. If you couldn't quite follow what I discussed, then don't feel too bad, because I'm sure that you'll be able to grasp the concepts after a while. In any case, you're probably itching to do some sort of programming in assembly language. Well, I'm afraid you are just going to have to wait some more, because we're only going to start programming next month. This month I've got to take you through a very important concept in assembly language—registers and addressing modes.

Registers

Perhaps one of the most important parts of your computer besides the microprocessor is something called a register. A register can be defined as really nothing but a "special storage location", to quote the VIC-20 Programmer's Reference Guide. And though it doesn't sound too impressive a definition, it is quite an accurate description of what a register is. But what we want to concentrate on is the word "special." What makes these registers so different from other memory locations? What's the use of having them? Why all the fuss over them? Those are probably some of the questions you're asking yourself—and with good reason. The answer to all of these is as follows: registers are important because they allow the microprocessor to keep track of where it's at and what it's doing. They are only different from other memory locations in that they are reserved for access by the microprocessor itself. You can mess around with them, but if you're not careful you can very well crash the system (it's unlikely, though). That's why all the fuss over them—you use them to "manipulate" the microprocessor. These registers are the only ways in which the microprocessor can communicate with you and thus using them to your advantage can sometimes do wonderful things for you (sometimes . . .). So is it clear now? Now do you see why they are so important? If not, then you had better read on, because now I'm going to describe the registers to you.

The Accumulator: This is probably the most important register from our point of view. Almost everything the microprocessor does uses the accumulator (the "acc.") in some way, and for good reason. It has the most machine language instructions to use and affect it, including the only math instructions. I have yet to see a machine language program that doesn't use it in some way.

The X and Y Registers: Often called the "index" registers, these are used mainly as indexes or offsets and as temporary storage while the accumulator is busy doing something else. They both have most of the instructions as the accumulator, excepting the math instructions, as well as a few unique ones of their own. It's also rare to find a program that doesn't use one of these, if not both.

The Status Register: This register really isn't the same as the preceding three in that it isn't used to move or change data around, but instead uses each of its bits (except one) to act as flags for certain conditions (the other bit is simply not used). A bit may be set in this register when only one specific condition occurs. This allows you to make comparisons and then use the status register to decide what to do next. The updating of the status register is automatic, and so you really don't worry about it too much. Quite a few instructions use or affect it. (More on the status register when we get to part V—Branches and Comparisons)

The Stack Pointer: This register is used as a pointer to the next free location on the Stack. It isn't of much use now to you, since you probably don't know what the Stack is, so I won't go any further in detail until we get to part VI—Sub-routines and the Stack. (If you really want to know what the Stack is, refer to page 133 of the Programmer's Reference Guide.)

The Program Counter: This is the "master" register and also the biggest one. Whereas all the other registers are only 8 bits long (one byte), this register is 16 bits long (two bytes). You see, the program counter, PC for short, contains the address of the instruction currently being executed. And if you remember last month's article, the last memory address in the VIC or any PET/CBM is $FFFF, or 65535 decimal, which cannot fit in one byte, no matter which way you try. But it will fit in two bytes (exactly), and that is the reason for having the PC two bytes long. Otherwise it couldn't keep track of where the computer is. I said that it could be considered the "master" register. This is because it determines where the microprocessor will get its next instruction from, sort of like BASIC linenumbers. Thus you could say that it controls the chip, because it tells it where it will get the next byte of data. The only instructions that affect it directly are the branch, jump and jump-to-subroutine commands. (More on that later on).

Well, there you have it. I've just described to you the functions of each register and why they are important. I hope that it's evident why registers form an integral part of machine- and assembly-language programming. If you're really stuck, though, I suggest you get some kind of book on machine or assembly language that will explain to you in detail all that you'd like to

know about the registers and their importance.

Addressing Modes

As you'll soon learn, each instruction in assembly language has what is called an **addressing mode.** "What is that," you say? It is simply the way in which the instruction currently being executed by the microprocessor will get or store its data. This can also be done in BASIC. For example, the line 10 A = PEEK (309 + X*Z) -1 will give the variable A whatever value is found at 309 + X*Z and then add one to that figure. You see, here we are giving it information as to where to find the data it needs, and so we are giving it an addressing mode, which in this case is 309 + X*Z. Although it doesn't operate quite like that in assembly language, I think it gives you an idea as to how an addressing mode operates. To further demonstrate it, I'm going to give you examples of all the addressing modes available using one of the most common assembly language instructions—LDA. Can you guess what it means? (Remember how I said that assembly language was composed of *mnemonics,* or memory joggers?) If you said "LoaD the Accumulator" then you were right! This instruction (which we'll learn more about next month) loads or places a new value in the accumulator, sort of like POKEing it. In any case, it can be used in a lot of different addressing modes, which go as follows:

Zero Page: This mode will take the data from page zero (the first 256 bytes from $00 to $FF) and place it in the accumulator. **Example:** LDA $06—load the accumulator with the value found in byte $06.

Absolute: This is the same as Zero Page, but instead loads the acc. from the address specified by the two bytes following the instruction. **Example:** LDA $1F2A—will load the acc. with the value in byte $1F2A (7978 dec.)

Immediate: Probably the simplest, this loads the acc. with the value of the byte following it, which you have already defined. To separate it from Zero Page mode, the pound sign '#' precedes the byte. **Example:** LDA #$09—will place the value $09 in the accumulator.

Zero Page indexed with X: Loads the acc. with the value of the byte following it **plus** the value in the X register and then loads the acc. from the address obtained by the addition of both registers. **Example:** LDA $1F,X (where X = $05 - will load the acc. from the byte found at $1F + $05, or $24.

Indexed Indirect: This one is complicated, as it will load the acc. from the address which it determines by first taking the address you specified, adding the X register, and then taking the two bytes it finds at that location to find the true address from which to load the accumulator (whew!) **Example:** LDA ($00,X) where X = $01 and where addresses $01 and $02 both contain $11. First the byte in the brackets is taken and added to the X register to equal $01 ($00 + $01). Then it takes the value it finds at $01 and $02 (addresses have to be two bytes long), puts them together (**not add**) to equal $1111 and then goes over to $1111 and places whatever value it finds there in the accumulator.

Indirect Indexed: This is sort of the same idea as indexed indirect, except it uses the Y register **outside** the brackets, which then means it calculates the address differently. First it takes the byte in brackets, goes over to that address and then calculates an address from the two bytes at that location and at that location plus one. It then goes over to this address, adds Y to it, and then loads the acc. from the byte at that address. **Example:** LDA ($05), Y where Y = $08, and $05 and $06 equal $13. The value is to be taken from the address at $05 and $06 **plus** Y. Thus the value will be taken from $1313 + $08, or $131B.

Absolute indexed by Y or X: This is the same as absolute except that it will add the value of one of the registers (which you specify) to the two-byte address that follows the instruction. **Example:** LDA $5000,Y where Y = $07 - will load the acc. from $5000 + $07, or $5007.

Implied: This mode isn't really a mode since there are no addresses to be calculated. Instead the instruction simply executes, just as long as it doesn't need any address to execute. To show you this, I'll have to introduce a new instruction: NOP, which stands for No-OPeration. All it does is cause the microprocessor to wait a few microseconds and then continue on with its work. **Example:** NOP (that's it!)

Well, that's basically all the addressing modes there are for the 6502. There is still another one that affects the JMP (jump) instruction, but I'll leave that for later. In any case, you can see that there are a lot of addressing modes to use, and though some may seem confusing now, they'll become clear as we use them in our explorations in the coming months.

Next Month . . .

Next month we are going to actually start some programming, as I'll give you some examples to use on your computer. Not only will they be interesting (at least, after a while), they will also clear up and reinforce all that you've learned so far in this column. If you want to participate, you'll need some form of monitor. I recommend the Vic MON cartridge, but as long as you've got some kind of monitor you should be O.K. So until next month, happy computing!

COMPU SENSE

VIC-20®

VIC-20®	**Personal Computer**	$179.95
VIC-1011A	**RS232C Interface**	39.95
VIC-1515	**Printer**	334.95
VIC-1530	**Datasette**	67.50
VIC-1540	**Disk Drive**	349.95
VIC-1010	**Expansion Module**	139.95
VIC-1311	**Joystick**	9.95
VIC-1312	**Game Paddles**	19.95
VIC-1600	**Telephone Modem**	99.95
CM-151	**Terminal 40**	29.95

Produces 40 column output of information received through the modem

VIC-1210	**VIC 3K Memory Expander Cartridge**	34.95

Plugs directly into the VIC's expansion port. Expands to 8K RAM total.

VIC-1110	**VIC 8K Memory Expander Cartridge**	52.50

8K RAM expansion cartridge plugs directly into the VIC.

VIC-1011A	**RS232C Terminal Interface**	39.95

Provides interface between the VIC-20 and RS232 telecommunications modems. Connects to VIC's user port

CM	**16K Memory Expander**	89.95
CM	**24K Memory Expander**	110.95

CARDBOARD 6 $87.95
An expansion interface for the VIC-20. Allows expansion to 40K or accepts up to six games. May be daisy chained for more versatility.

CARDBOARD 3 $29.95
Economy expansion interface for the VIC-20.

CARD "?" CARD/PRINT $79.95
Universal Centronics Parallel Printer Interface for the VIC-20 or CBM-64. Use an Epson MX-80 or OKIDATA or TANDY or just about any other.

CARDETTE $39.95
Use any standard cassette player/recorder with your VIC-20 or CBM-64.

CARDRITER $29.95
A light pen with six good programs to use with your VIC-20 or CBM-64.

BUSINESS & HOME APPLICATIONS FOR VIC-20®

CW-107A	**Home Calculation Program Pack**	$48.95
CPV-31	**Data Files** - your storage is unlimited	14.95
CPV-96	**Household Finance Package** - to keep records of all your household expenses	30.95
CPV-208	**Bar-Chart** - display your numerical data	8.95
CH	**Turtle Graphics** - learn programming	34.95
CH	**VIC Forth** - is a powerful language for BASIC programming	49.95
CH	**HES MON** - is a 6502 machine language monitor with a mini-assembler	34.95
CH	**HES Writer** - time-saving word processing tool	34.95
CH	**Encoder** - keep your personal records away from prying eyes	34.95
CT-21	**Statistics Sadistics** - statistical analysis	14.95
CT-121	**Total Time Manager 2.0** - creates personal or business schedules	15.95
CT-124	**Totl Label** - a mailing list and label program	13.95
CT-125	**Totl Text BASIC**	15.95
CT-126	**Research Assistant** - keep track of reference data	17.50
CT-140	**Totl Text Enhanced**	29.95
CM-152	**Grafix Designer** - design graphic characters	12.95
CQ-5	**Minimon** - allows you to program, load, save, or execute machine language programs	13.95
CT-3	**Order Tracker**	15.95
CT-4	**Business Inventory** - to maintain record of inventory	15.95
CPV-210	**Bidder**	13.95
CPV-217	**Cash Flow Model** - determine cash flow	12.95
CPV-220	**Client Tickler**	16.95
CPV-221	**Club Lister**	13.95
CPV-224	**Depreciator**	9.95
CPV-236	**Investment Analyst** - keep track of investments and investment opportunities	12.95
CPV-251	**Present Value**	10.95
CPV-269	**Super Broker**	12.95
CPV-270	**Syndicator** - calculates whether to buy or sell	13.95
CPV-274	**Ticker Tape** - maintains investments profile	14.95
CPV-276	**Un-Word Processor** - screen editor	16.95
CPV-286	**Phone Directory** - never lose a phone number again	9.95
CS-111	**Checkbook** - home "utility" program	14.95
CPV-294	**Calendar My Appointments** - print a calendar for every month in any year.	14.95
CPV-296	**The Budgeter** - place your personal finances in order	12.95
CPV-327	**HESCOM** - transfers data and programs bidirectionally between VICs at three times the speed of a disk drive	40.95
CPV-328	**HESCOUNT** - monitors program execution	19.95
CHV	**HESPLOT** - Hi-res graphics subroutines	12.95
CPV-367	**Conversions** - figures, volume, length, weight, area, and velocity to all possible configurations	7.95
CC	**The Mail** - your complete mail program Cassette	24.95
	Disk	29.95
CS	**Home Inventory** - lists your home belongings	17.95
CS	**Check Minder** - (V-20 & 64) keep your checkbook the right way	14.95
CS	**General Ledger** - a complete general ledger	19.95

BUSINESS & HOME APPLICATIONS FOR C-64

CHC-504	**HES Writer** - word processor	$39.95
CHC-503	**Turtle Graphics II** - utilizes the full graphics of your 64	49.95
CHC-502	**HESMON** - machine language monitor w/mini-assembler	34.95
CHP-102	**6502 Professional Development System**	29.95
CFC	**Data Files** - a management program	27.95

MANY MORE PROGRAMS FOR YOUR 64 & 20

COMMODORE SOFTWARE

VIC-1211A **VIC-20 Super Expander** $57.99
Everything Commodore could pack into one cartridge - 3K RAM memory expansion, high resolution graphics plotting, color, paint and sound commands. Graphic, text, multicolor and music modes. 1024x1024 dot screen plotting. All commands may be typed as new BASIC commands or accessed by hitting one of the VIC's special function keys. Includes tutorial instruction book. Excellent for all programming levels.

VIC-1212 **Programmer's Aid Cartridge** $45.99
More than 20 new BASIC commands help new and experienced programmers renumber, trace and edit BASIC programs. Trace any program line-by-line as it executes, pause to edit. Special KEY command lets programmers redefine function keys as BASIC commands, subroutines or new commands.

VIC-1213 **VICMON Machine Language Monitor** $48.99
Helps machine code programmers write fast efficient 6502 assembly language programs. Includes one line assembler/disassembler.

GAMES FOR YOUR VIC-20®

CCS	**Cribbage**	$14.95
CCD	**Motor Mouse**	12.99
CW-1901	**Avenger Cart.** - an invasion of space intruders and you're the VIC "Avenger"	24.95
CW-1904	**Superslot Cart.** - great music and sound effects!	24.95
CW-1906	**Super Alien Cart.** - you're trapped in a maze	24.95
CW-1907	**Jupiter Lander Cart.** - pilot your "Jupiter Lander"	24.95
CW-1908	**Draw Poker Cart.**	24.95
CW-1909	**Midnight Drive Cart.** - authentic night driving	24.95
CW-1910	**Radar Rat Race**	24.95
CW-1911	**Sky Falling**	24.95
CW-1912	**Mole Attack** - a colorful "cartoon action" game	24.95
CW-1913	**Raid on Ft. Knox** - try to escape the guards	24.95
CW-1914	**Adventure Land** - Formerly available only on larger, more expensive computers. All Adventure games are decoded to "talk" on the Type N Talk voice synthesizer (available from VOTRAX)	31.95
CW-1915	**Pirate Cove Adventure** - Yo, ho, ho, & a bottle of rum	31.95
CW-1916	**Mission Impossible Adventure**	31.95
CW-1917	**The Count Adventure** - trapped in Dracula's castle with 3 days to find and destroy the vampire	31.95
CW-1918	**Voodoo Castle Adventure** - you have to free Count Yorga from a curse	31.95
CW-1919	**Sargon II Chess** - seven challenging play levels	31.95
CW-1923	**Gorf** - (The smash-hit arcade game!)	31.95
CW-1924	**Omega Race** - the ultimate space game	31.95
CW-1937	**Seawolf** — an explosive Bally Midway arcade "classic"	24.95
CH-G202	**Maze of Mikor** - adventure-packed game with stunning graphics	15.95
CH-G203	**Tank Wars**	15.95
CH-G205	**Pinball**	13.45
CH-G206	**Simon** - It gets tougher as you get better. Great for kids of all ages.	13.45
CH-G207	**Fuel Pirates**	13.45
CH-G209	**Laser Blitz**	15.95
CH-G210	**Tank Trap**	15.95
CH-G211	**Concentration**	13.45
CH-G212	**Dam Bomber** - pilot your plane, avoid enemy fire	13.45
CH-C307	**Shamus** - search room after room for the shadow-eluding androids, two levels of intense arcade action	34.95
CH-C308	**Protector**	36.95
CPU-79	**Breakout**	7.95
CPU-85	**Hangman** - unbelievable graphics and sound	9.95
CPU-87	**Memory** - VIC challenges your memory	9.95
CPU-88	**Match** - hand and eye coordination	7.95
CPU-89	**Monks** - a devilish game of logic	7.95
CPU-108	**Bomber** - you must decide who you want to fly for, then pick a target and your experience level	9.95
CPU-109	**Amok** - the halls of Amok are populated by robots that obey one instruction - get the intruder	20.95
CPU-153	**Tank vs. UFO** - the tank is moving back and forth along the base, shoot the UFO before it shoots you	9.95
CPU-194	**Snakman** - Pacman for the VIC	14.95
	Defender on Tri - you're the pilot of the experimental ship, Defender	17.95
	3-D Man - the popular arcade game, requires 3K	17.95
	Exterminator - a game full of bugs	20.95

GAMES FOR YOUR 64

CCS	**Cribbage**		$17.95
CFC	**Flight 64** - what a program!	Cassette	14.95
		Disk	16.95
CFC	**Spright Generator**	Cassette	15.95
		Disk	17.95
	Mastermind (VIC-20) $12.95	(C-64)	19.95
	Star Trek (VIC)		9.95
	Black Jack (VIC)		11.95
	Tic-Tac-Toe		7.95
	Backgammon		14.95
	Maze 64		15.95

Prices subject to change.

TO ORDER:
P.O. Box 18765
Wichita, KS 67218
(316) 684-4660

MasterCard VISA

Personal checks accepted (Allow 3 weeks)
or C.O.D. (Add $2) *Handling charges $2.00*

VIC-20® is a registered trademark of Commodore

RAVINGS OF A MADMAN

by Tim Parker
Ontario, Canada

Last month I took a few lines to look at quicky fingersavers for the VIC-20. I promised to pass on a couple of tricks for protecting your programs this month, and being a (somewhat) honest type, I'll do exactly that, after a slight diversion.

Unfortunately, there is really no way to protect a program in absolute terms. If it can be written, it can be figured out. Ask any Apple II owner how he gets into protected program disks, and he'll probably respond that he uses one of the commercially available routines that decode protection circuits. Two spring to mind, namely "Locksmith," and "Quick and Dirty." Their programmers wrote them to allow "cracking" of protection routines. And they work—very well!

But such a program doesn't exist as such for the VICs, PETs, or other similar computers (although the chances of one floating around that I don't know about are quite large). The primary reason for this is that all Commodore computer programs can be accessed very easily by a knowledgable programmer, simply because the Commodore's operating systems are different from Apple's. (I know of several people who have even downloaded every game cartridge available for the VICs, and possess large libraries of "bootleg" disks.)

This of course brings up the inevitable question of ethics. Is it legal (or fair) to duplicate a program without paying for it? The legal answer is rather straightforward: NO! It is in violation of copyright laws to duplicate a program, except as a backup. Note, however, that few, if any, suits have been won on this issue. The laws in most countries do not directly address computer software.

As for whether it is fair, that is a dangerous question. It can perhaps be answered in a round-about way by considering that if a program you would never ordinarily buy is offered to you simply to look at for curiosity's sake, then who gets hurt? The owners of the copyright would not make any money from you anyway, as you wouldn't have forked over the megabucks required to purchase the thing in the first place. But if you find the program useful, should you then pay for it? These, gentle reader, are questions I cannot answer: if I did, I would get into trouble, as every programmer feels differently.

I can pass on the overwhelming opinion of many of the higher level computer users (of which I classify myself one). Software costs a bundle these days. A PASCAL program can cost from three hundred bucks up. The average software price in CP/M systems is about six hundred. These are overinflated, without a doubt. And that is definitely why there is such a problem with software piracy. If my friends had bought every piece of software they owned, they would have been bankrupt very quickly. The answer is simple: lower software prices. As an example, there is one company that now offers a CP/M PASCAL (a good one, as it happens) for thirty bucks! They feel that too much was being charged before, and that more people will feel like buying it not because of the low price, but because they can get a legal package for a reasonable price. I bought one of the first of their shipments, and after friends saw it, they decided that since thirty bucks wasn't much more than the price of copying the disks and photocopying the manual, they might as well order their own. And they did. The company picked up ten customers they wouldn't have had at the higher price.

Such a problem, although not so rampant, does exist in Commodore land. Games, especially, are somewhat on the high side. One manufacturer sent me his brochure, and the games were on the far side of forty bucks each. Now for a game I haven't even seen, that seems like a big gamble to me. (By the way, have you noticed the pictures in the ads and on the game boxes never reflect the game itself?)

What can we do about it? Not much. But we can be more selective in what we buy. Don't throw good cash on an untried game or program. Check it out first! As for copying a friend's programs, only you can decide whether to do it or not. But remember . . . it's not ethical! I doubt anyone will lose too much sleep about it though.

After that diatribe, I get back to where I wanted to be: on protecting programs. The Commodore, as noted above, cannot be completely protected. After a program is loaded into memory, anyone can examine it as long as it hasn't been RUN. The reason is that certain memory locations that control the RUN/STOP key, LIST and SAVE functions can be set to disable, but only during a RUN. (Actually they can be done before the RUN, but who is going to disable a LIST of a program they are trying to copy?)

The way to prevent a LIST of the program being made is to add the command POKE 755,200 to your program. To undo the command, POKE 755,199.

A look at memory maps for the VIC, PET or whatever will show a few other tricks. For example, the VIC-20 employs memory locations 818 and 819 for the SAVE functions (816 and 817 are for LOAD). Thus, POKEing 818 to 165 will kill the SAVE to an extent . . but the RUN/STOP and RESTORE keys must also be deactivated. The routine to prevent SAVEing of your programs is POKE 818,165:POKE 802,0:POKE 803,0. (Locations 802 and 803 control the in-

put/output restore function.) The command is revoke by POKE 818,133:POKE 802,243:POKE 803,243.

A study of memory maps will reveal other tricks . . . more will be reported in upcoming columns. (Memory maps are published in several different magazines and books. Dig for them!)

Last month I mentioned video expansion boards for the VIC-20 to give 40, 60 and 80 columns. Some of these are advertised in the magazines with extravagant ads, some of which I feel are slightly misleading. One thing to understand from the start: the video image will be different. Although some boards utilize a matrix of 5 x 7 (or better), when projected onto a television, there is inevitable picture disintegration. (If you happen to own a monitor . . . don't worry about it!)

Also, most boards do not offer color. That is not a problem for those people who buy VIC-20s for business use (but then why did they buy a VIC-20?) or for the occasional telecommunication use. But games on an expanded screen are a wipeout when the usual VIC-20 programs are run. Certainly, the PET games can be used, and some boards will apparently support VIC-64 games, but don't buy without checking them out first. Being stuck with a board that costs several hundred dollars and sits unused is very frustrating.

This doesn't mean not to look at them. Programming with even a forty character screen is so much easier on the eyes than the VICs twenty-odd. Programs actually become readable. That alone may be worth the purchase price to some who do a lot of development work. As for POKEing screen locations and colors, ask the dealer or manufacturer. Some will have an entirely new screen map, while some don't allow individual character POKEing. That makes game design near impossible (and that is, after all, very important).

Speaking of games, after sending off last month's column on games, a new catalog arrived from United Microware Industries (UMI). (See ad in this month's issue for address—ed.) They have added a few new products to their line. Write for their color brochure.

A relatively new game that has attracted a lot of attention is Super Paratrooper, by Nic Dudzik. (Protecto is one distributor: they had a sale price on it last I heard, although that may have expired. Get in touch with them for details.) The graphics on the copy I saw were very well done. They compare very favourably to those on the Atari, although there are the subtle differences of not having player/missile graphics.

As could have been expected, a whole series of arcade game look alikes for the VIC have been appearing. Some are well done, while others seem best left in their boxes. Again, the old advice applies: look before you buy! If your dealer will not give you a demo, there must be some reason why. Mail order is a different problem entirely. The key is to make sure the company will back their products. Some are even giving a money back guarantee if the products are not up to your satisfaction. Personally, I'd rather pay a buck or two extra to get that kind of commitment, than save pennies and be stuck with a shelf load of garbage.

Shop around, also. There are a few dealers engaged in the old price war game. All that is to the benefit of the consumer, but be wary of extravagant claims. You simply cannot get fifty different games on one tape for ten bucks. (Although, if anyone offers that, let me know, huh?)

(Editor's note—We all acknowledge the high price of some software, but must also accept the fact that software piracy (copying) is rampant. Most software manufacturers raise their prices to compensate for high losses due to theft (copying). As more companies produce quality software, competition will force prices down. Copying software is unethical and illegal—just because other people do it does not make it legal.)

GOBBLE!

by Tim Parker
Ontario, Canada

Gobble! is relatively easy to play, but is not such a simplistic game that interest quickly fades. It is a game ideally suited to the screen size of a VIC-20, although it can easily be modified to the larger size of the VIC-64, PET, or other computers. It fits inside the memory of an unexpanded VIC-20.

The game is based upon a maze generation program that draws a unique maze (i.e. has one entrance and one exit). Several people have developed the algorithms required for this purpose, including David Matuszek (*BYTE*, December 1981, Vol 6 No 12) and Charles Bond (*COMPUTE!*, December 1981, Vol 3 No 12). The latter has been adapted for use in this program, although most variables had to be changed to accomodate the screen size. The details of the maze generator segment will be left for the reader to investigate.

The object of the game, simply stated, is to "eat" as many dots as possible with your character (a ball) while avoiding the enemy (a square) that runs the maze at the same time as you, albeit slower. For each dot you eat, you get one point. For each dot the enemy eats, you lose one point.

In the early levels of the game, there are several cross connections provided to make escaping the enemy easy. As the levels are completed, the number of connections becomes fewer, and the game subsequently more difficult.

Bonus points are awarded on completion of the fourth, ninth, and fourteenth rounds (assuming you get that far). A player may exit the maze through the hole in the bottom at any time, whether all the dots are eaten or not. The next level will then be generated.

The version printed here is intended for keyboard control. Modification of joystick is easy, and the details are

JOYSTICK MODIFICATIONS

Add the following lines:

```
9000 DD=37154: P1=37151: P2=37152
9010 POKE DD, 127: P=PEEK(2)AND128: jO=-(P=O)
9020 POKE DD,255: P=PEEK(P1)
9030 J1=([PAND8]=-0): J2=-([PAND16]=0): J3=-
     ([PAND4]=0): RETURN
90   GOSUB 9000
100  IFJQANDPEEK(L+1)<> 160THENPOKEL,32:L=L+1
     :GOSUB600: POKEL,81
110  IFJ2ANDPEEK(L-1)<> 160THENPOKEL,32:L=L-1
     :GOSUB600: POKEL,81
120  IFJ1ANDPEEK(L+22)<>160THENPOKEL,32:L=L+22
     :GOSUB600: POKEL,81
130  IFJ3ANDPEEK(L-22)<> 160THENPOKEL,32:L+L-22
     :GOSUB600: POKEL,81
```

```
10 REM TIM PARKER...GOBBLE!
15 PRINT"{CLEAR}"
20 POKE36879,25:V=36878:S0=36875:PT=0:D
L=0
30 POKEV,10
40 GOSUB6000
50 L=INT(RND(-TI)):DEFFNA(X)=INT(RND(1)
*X)+1
60 DL=DL+1:IFDL=50RDL=100RDL=15THENGOSU
B5000
65 GOSUB1000
70 L=7734:POKEL,81:POKEL-22,160
80 GOSUB2000
90 GETA$:IFA$<>""THENB$=A$
100 IFB$="K"ANDPEEK(L+1)<>160THENPOKEL,
32:L=L+1:GOSUB600:POKEL,81
110 IFB$="J"ANDPEEK(L-1)<>160THENPOKEL,
32:L=L-1:GOSUB600:POKEL,81
120 IFB$="M"ANDPEEK(L+22)<>160THENPOKEL
,32:L=L+22:GOSUB600:POKEL,81
130 IFB$="I"ANDPEEK(L-22)<>160THENPOKEL
,32:L=L-22:GOSUB600:POKEL,81
140 IFL+1=A0RL-1=A0RL+22=A0RL-22=ATHEN8
000
150 IFL=8152THEN60
160 POKES0,0
170 GOSUB2010
```

given at the end of this article. The joystick version runs approximately twice as slow as the keyboard, due to the frequent subroutines required for joystick commands.

Instructions are given at the beginning of the game, after an introductory title and short tune.

The program is constructed in a series of subroutines to allow modifications to be easily added. The routines are explained below.

The maze generation section is lines 1000-1999. The color of the background is controlled by variable CL. Lines 1210-1230 add cross connections at the lower difficulty levels, controlled by variable DL.

The enemy is moved by lines 2000-2999. The movement is executed in line 2010. Lines 2500-2510 subtract one point from the score for every dot the enemy eats.

Lines 5000-5999 control the introductory title and jingle. The prompt for instructions is given.

Instructions are in lines 7000-7999.

The game termination sequence is given in lines 8000-8999.

The major control loop is at lines 60-160. Here the enemy is controlled, and the keyboard input obtained and analyzed. A joystick branch to lines 9000-9999 is used here for joystick control.

The meaning of most of the variables should be obvious from their context.

Strategy in Gobble! will become obvious after a few games. Study of the enemy's movement will reveal a very useful fact about the direction it takes. You may find it is not always a good idea to clean out a level before moving to the next.

Only one "life" has been used here, although more could be added. Also only one enemy has been added. At higher levels, more can be introduced, although the game slows down considerably if a large number are controlled.

Good luck, and start Gobbling!

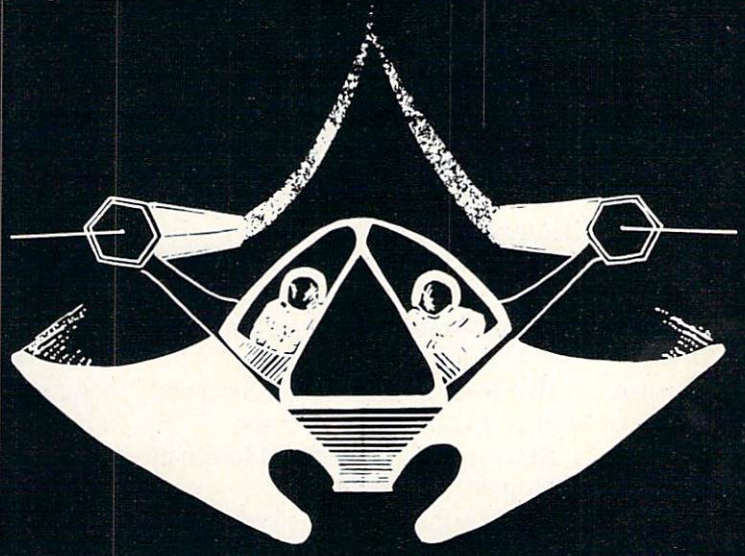
```
500 GOTO90
600 IFPEEK(L)=46THENPT=PT+1:POKESO,231
610 RETURN
1000 A(0)=2:A(1)=-44:A(2)=-2:A(3)=44
1001 CL=CL+1
1002 IFCL=1THENPRINT"{BLACK}"
1003 IFCL=2THENPRINT"{RED}"
1004 IFCL=3THENPRINT"{CYAN}"
1005 IFCL=4THENPRINT"{PURPLE}"
1006 IFCL=5THENPRINT"{GREEN}"
1007 IFCL=6THENPRINT"{BLUE}"
1008 IFCL=7THENPRINT"{YELLOW}":CL=0
1010 WL=160:HL=46:SC=7690:A=SC
1020 PRINT"{CLEAR}"
1030 FORI=1TO21
1040 PRINTSPC(1)"{REV}
   "
1050 NEXTI
1060 POKEA,4
1070 POKESO,143:K=FNA(4)-1:X=K
1080 B=A+A(K):POKESO,0:IFPEEK(B)=WLTHEN
POKEB,K:POKEA+A(K)/2,HL:A=B:GOTO1070
1090 K=(K+1)*-(K<3):IFK<>XTHEN1080
1100 K=PEEK(A):POKEA,HL:IFK<4THENA=A-A(
K):GOTO1070
1200 POKE8152,32:POKE8130,32
1210 FORZ=1TO(10-DL):X=FNA(16):Y=FNA(18
)
1220 POKE7704+X+Y*22,46
1230 NEXT
1240 POKE8174,160
1250 PRINT"{HOME}SCORE=";PT;"{HOME}";SP
C(12)"HI=";HS
1300 POKE7788,140:POKE7832,133:POKE7876
,150:POKE7920,133:POKE7964,140
1310 IFDL<10THENPOKE8030,DL+176
1320 IFDL>9THENS1=INT(DL/10):POKE8030,S
1+176:POKE8052,DL+176-S1*10
1500 RETURN
2000 A=8152:POKEA,102:K=2
2010 B=A+A(K)/2:IFPEEK(B)<>160THENGOSUB
2500:POKEB,102:POKEA,32:A=B:K=(K+2)+4*
(K>1)
2030 K=(K-1)-4*(K=0)
2040 PRINT"{HOME}SCORE=";PT;"{LEFT} "
2050 RETURN
2500 IFPEEK(B)=46THENPT=PT-1:IFPT<0THEN
PT=0
2510 RETURN
5000 PRINT"{CLEAR}{DOWN}{DOWN}{DOWN}
{DOWN}{DOWN}{DOWN}";SPC(5)"{GREEN}
{REV}BONUS SCORE"
5030 PRINT"{DOWN}{DOWN}{DOWN}{DOWN}
```

```
{DOWN}  500 POINTS BONUS!"
5040 PT=PT+500
5050 PRINT"{HOME}SCORE=";PT
5060 FORWL=1TO100:POKES0,INT(RND(1)*128
)+128
5070 FORT=1TO10:NEXT:NEXT
5080 POKES0,0
5090 FORT=1TO1000:NEXT
5160 RETURN
6000 PRINT"{CLEAR}{DOWN}{DOWN}{DOWN}
{DOWN}{DOWN}";SPC(5);"{GREEN}*********
**"
6002 PRINTSPC(5)"*          *"
6004 PRINTSPC(5)"* GOBBLE! *"
6006 PRINTSPC(5)"*          *"
6008 PRINTSPC(5)"**********"
6009 GOTO6500
6010 PRINT"{DOWN}{DOWN}{DOWN}{DOWN}
{RIGHT}{RIGHT}{RIGHT}{RIGHT}{CYAN}INST
RUCTIONS?"
6060 GETA$:IFA$=""THEN6060
6070 IFA$="Y"THEN7000
6120 RETURN
6500 FORWL=1TO3
6505 FORSC=1TO9
6510 READX:POKES0,X
6520 FORT=1TO100:NEXT
6530 NEXT:RESTORE:NEXT
6535 POKES0,219
6536 FORQ=10TO0STEP-1:POKEV,Q:FORT=1TO1
00:NEXT:NEXT
6540 POKEV,10:POKES0,0:GOTO6010
6560 DATA215,201,228,207,215,219,207,20
1,219
7000 PRINT"{CLEAR}          {REV}GOBBLE"
7010 PRINT"{DOWN}{BLACK}    A RANDOM MAZ
E WILL"
7020 PRINT"BE DRAWN ON THE SCREEN";
7030 PRINT"CONSISTING OF DOTS."
7040 PRINT"{CYAN}  ON THE SCREEN, YOU"
7050 PRINT"ARE SHOWN AS q, AND"
7060 PRINT"YOUR TASK IS TO EAT AS";
7070 PRINT"MANY DOTS AS YOU CAN."
7080 PRINT"YOU SCORE ONE POINT"
7090 PRINT"FOR EACH DOT EATEN."
7110 PRINT" {PURPLE} ALSO ON THE SCREEN
"
7120 PRINT"IS YOUR NEMESIS, SHOWN";
7130 PRINT"AS &. THIS TRAVELS"
7140 PRINT"THROUGH THE MAZE, ALSO";
7150 PRINT"EATING DOTS. YOU LOSE"
7160 PRINT"ONE POINT FOR EACH DOT";
7170 PRINT"IT EATS."
```

```
7180 PRINT" {GREEN} IF THE & GETS CLOSE
"
7190 PRINT"TO YOU, IT WILL EAT"
7200 PRINT"YOU, AND YOU LOSE.{DOWN}"
7210 PRINTSPC(12)"{REV}HIT A KEY{OFF}
";
7220 GETA$:IFA$=""THEN7220
7230 PRINT"{CLEAR}  THERE IS AN ENTRANC
E";
7240 PRINT"TO THE MAZE THAT SHUTS";
7250 PRINT"AFTER YOU ENTER. ONLY"
7260 PRINT"ONE EXIT EXISTS. TO"
7270 PRINT"LEAVE THE MAZE, AND GO";
7280 PRINT"TO THE NEXT LEVEL, YOU";
7290 PRINT"MOVE INTO THE EXIT."
7300 PRINT"AS THE LEVELS INCREASE";
7310 PRINT"SO DOES THE DIFFICULTY";
7320 PRINT"OF THE MAZE."
7330 PRINT"{BLACK}BONUS POINTS ARE GIVE
N";
7340 PRINT"AT LEVELS 5 10 AND 15."
7345 PRINT"{BLUE}{DOWN}";SPC(10);"I=UP"
7346 PRINTSPC(5)"J=LEFT   K=RIGHT"
7348 PRINTSPC(10)"M=DOWN"
7350 PRINT"{DOWN}{CYAN}TRY FOR A HIGH S
CORE."
7360 PRINT"IT'S NOT THAT EASY...."
7370 PRINT"{DOWN} {RED} {REV}HIT A KET
TO START{OFF}";
7380 GETA$:IFA$=""THEN7380
7390 RETURN
8000 POKEL,102:POKEA,32:CL=0:POKES0,0
8010 PRINT"{HOME}       GAME OVER    "
8015 IFPT>HSTHENHS=PT
8020 FORT=1TO1000:NEXT
8030 PRINT"{CLEAR}{UP}{UP}{UP}   ANOTHE
R GAME?"
8040 GETA$:IFA$=""THEN8040
8050 IFA$<>"Y"THENEND
8090 PT=0:DL=0
8100 GOTO60
```

To save wear and tear on the fingers, a copy of this game can be obtained by sending $3 (to cover first class postage and duplicating), a mailer, and a blank cassette to:
Tim Parker, 66 McKitrick Dr., Kanata, Ontario, K2L 1T7.

GAME PROGRAM DEVELOPMENT KIT



for the

VIC - 20 and COMMODORE - 64

Registered trademarks of Commodore Business Machines, Inc.

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

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

DESIGNED TO RUN ON ALL VIC-20's

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

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

French Silk smooth ware

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

507-263-4821

PET/CBM

A Super Editor for the PET

by F. Arthur Cochrane
Beech Island, SC

For those who may not know what Basic—Aid is I will start with a little background. Basic-Aid is a BASIC program development tool for the PET and was originally written by Bill Seiler of Commodore and is very much like the Toolkit. It has the following commands;

Aid—A Help function when a BASIC program error occurs.

Auto—Auto line numbers for program entry.

Break—Break to the TIM machine language monitor in the PET.

Change—Search for an old string and replace it with a new string in a BASIC program.

Delete—Delete a range of lines from a BASIC program.

Find—Find a string in a BASIC program and print the lines where it occurs.

Kill—Disable Basic-Aid from use.

Number—Renumber a BASIC program correcting all GOTOs and GOSUBs.

Repeat—Enable repeat keys.

Trace—Enable the trace function, which prints the line number and token in a window when a program is run.

The program was a 2K program which loaded into the top 4K of a 32K PET and worked only on Upgrade BASIC.

The next version of Basic-Aid that I know about was a version from Commodore Canada. This version was upgraded for BASIC 4.0 and added the following commands;

First—List a BASIC program directly from the disk to the screen.

Hex—Convert HEX to decimal and decimal to HEX.

Lower—Put the PET into lower case.

Merge—Merge a program from the disk with the one in memory.

Read—Read a sequential file directly from the disk to the screen.

Start—Print the loading address of a program on the disk.

Upper—Put the PET into upper case.

The next version of Basic-Aid that I came across had the following commands and functions added:

Dump—Dump the variables defined in the program.

Crt—Dump the screen to the printer.

Pack—Remove the extra spaces and REM's from a BASIC program.

Dos—Also the DOS Support commands (@,>,/,↑) were included.

The ability to print the screen with the press of a key and to escape from the quote/insert mode were also added.

My additions to Basic-Aid have been the following commands and functions:

Size—Give the size of a program in memory or on the disk.

Spool—Send a file from the disk directly to the printer.

Un-new—Restore a program after a NEW.

The ability to scroll the BASIC program with the cursor control keys was added. The scroll feature was adopted from code for a version of the CBM assembler editor by Bill Seiler.

Many bugs were also fixed. I would like to thank Jim Butterfield for the AID4 program which allowed me to fix a renumber bug in Basic-Aid. The DOS commands also had bugs which were fixed. Also when upgraded to BASIC 4.0 the trace would not function because a compare was now incorrect. The screen dump was modified to allow printing to an ASCII printer.

Basic-Aid is a very powerful BASIC program development aid, but how does it compare to other programs available for the PET?

Basic-Aid has more features than the Toolkit and is more useful than a Toolkit alone.

The Disk-O-Pro has some useful features. The most important is the addition of BASIC 4.0 commands to Upgrade BASIC. Also the Print Using command for formated output is useful. The Disk-O-Pro will function with a Toolkit if one is present. A disadvantage is that the Disk-O-Pro must be in place for these commands to work in a program and it slows BASIC down. See **Compute** issue #8 page 112 for a complete review.

The Command-O adds the Print Using command, the Toolkit commands, and others to BASIC 4.0. The Renumber command is improved to allow renumbering in a line range instead of the whole program and the Trace function has been improved to show the whole line that is being traced. But again the Command-O must be enabled for the Print Using command and others to work in a program.

Power has some different commands also. It has the improved

Renumber command and a very powerful Trace function. It has a Search and Replace command with the option for don't care characters in the search string. Power also has instant keywords and instant subroutines options which can be useful. The XEC command is very powerful and has many options, such as merging a program from disk. Power has the option for other commands to be added to it. In fact there is a Power-Aid available from the user groups below. For a full review of Power see the Overview in **Compute** issue #18 page 136.

SYSRES is the newest BASIC utility on the market and has every command you can think of. It has commands to change with don't cares and at the start of lines, formated outputs to Commodore and ASCII printers, list program, sequential, and relative files from the disk. The GET and PUT commands allow editing sequential files. The thing I personally dislike is that you have to load SYSRES each time from the disk on reset or cold start. SYSRES can't be killed then SYSed to enable it, it must be reloaded from disk. Also SYSRES eats up 8K of memory. SYSRES can be loaded into RAM at $9000 and $A000 if the PET owner has RAM in the 2 sockets. I wish the author of SYSRES would sell a version of SYSRES in two 4K EPROMs so on cold start SYSRES could be called with a SYS.

So if you have a PET which Super-editor is for you? The answer will depend on the BASIC your PET has and the features you want a super-editor to have. Upgrade BASIC users can choose from the Toolkit with a Disk-O-Pro, Power, SYSRES, or Basic-Aid. Basic 4.0 users can choose the Toolkit, Command-O, Power, SYSRES, or Basic-Aid. Each super-editor has some features not included in the others. The user should get all the information on each and decide for himself. In this evaluation Basic-Aid has a strong selling point in that it is in the public domain and is FREE. There are other Super-editors not mentioned here but these are the ones most seen in ads and the ones with which the author is familiar.

Note that Original BASIC users are limited to a Toolkit only. Because of vast Zero page changes between Original BASIC and Upgrade BASIC and Original BASIC not being able to work with the Commodore disk, Basic-Aid as it stands now will not assemble for Original BASIC.

After Basic-Aid was done it was noticed that Basic-Aid was very similar to the Commodore Assembler Editor that is supplied with the Assembler Development Package. The only thing Basic-Aid lacked was a method to GET and PUT the source code files to the disk. Since the source code for the Commodore Editor is supplied with the Development Package it is available for easy modification and use.

The Assembler-Aid program changes the CHRGET Wedge routine to point to itself. A line that is input beginning with a line number is inserted into memory without being tokenized. Otherwise this program checks to see if the direct command is in its command list, and if it is the command is executed. If the command is not an editor command then a jump is made to Basic-Aid to let it check its commands.

Along with the GET and PUT commands the Format command was added to list out source code formated like the assembler formats. Also the Loader function was added so that the separate loader program is not needed unless the object code would load over the Assembler-Aid.

Using Basic-Aid as part of an assembler editor allows the programmer all the features of Basic-Aid to be used in the editing of matching language source code. The most powerful of these is the ability to scroll through the code with the cursor control keys.

But where do you get Basic-Aid and the Assembler-Aid programs? A PET user group is the best source, two user groups which can provide Basic-Aid are ATUG and TPUG (addresses below). For those who would like source code, Basic-Aid and Assembler-Aid source code in Carl Moser's MAE assembler format is available and there are programs available to convert MAE format files to CBM format files. Basic-Aid and Assembler-Aid can be assembled and burned into an EPROM and plugged into one of the empty sockets in the PET so it is available with a SYS and does not have to be loaded from disk each time the PET is reset or powered up.

In my 4032 PET I have Basic-Aid in the $9000 socket and the Assembler-Aid and Extramon in the $A000 socket. With these two EPROMs I only need the assembler program on disk to have a complete 6502 development system with editor, assembler, and debugger.

I hope that you will pass Basic-Aid and Assembler-Aid on to your friends. These programs are in the public domain and should be passed around freely. If anyone finds bugs or has comments please contact me about them. The command list for Basic-Aid and Assembler-Aid follows.

BASIC-AID COMMAND LIST

AUTO LINE NUMBERING

Syntax: AUTO inc.
AUTO

Auto prints the BASIC line numbers for you as you key-in a program. Enter AUTO with an increment, then enter the first line of the program. After hitting RETURN Basic-Aid will provide the next line number. Turn AUTO off by entering the command with no increment. Increment can be from 1 to 127.

BREAK TO THE MONITOR

Syntax: BREAK

The BREAK command calls the machine language monitor in the PET. This is a call to the monitor and not a break, so open files on BASIC 4.0 will remain open for printing from the monitor.

CHANGE TEXT

Syntax: CHANGE @ search string @ replacement string @, line range

The CHANGE command will search through a BASIC program for a string and change it to a replacement. The changed lines are displayed as they are changed. Because BASIC lines are tokenized it may be necessary to enclose the strings in quotes to change them. The whole program is searched unless a line range is given.

The line range has the same format as the list command.

DUMP THE SCREEN TO A PRINTER

Syntax: CRT

The screen is printed to a printer connected to the PET as device number 4. There are versions of Basic-Aid for ASCII and Commodore printers. The screen dump to a Commodore printer will be exactly like the screen. The screen dump to an ASCII printer will be in upper case only if the PET is in graphics mode or lower/upper case if the PET is in lower case mode.

DELETE A RANGE OF LINES

Syntax: DELETE line range

Deletes a given line range of lines from a BASIC program. The line range has the same format as the list command.

DISPLAY THE BASIC VARIABLES

Syntax: DUMP

DUMP will list the variables used in a BASIC program and their values. Dump does not list arrays. The variables are listed in the order in which they were created. The variables are printed in such a way that they can be edited with the screen editor so that a program can be stopped, the variables dumped, edited to new values, and the CONT command given to continue the program with the new values.

FIND TEXT

Syntax: FIND @search string@, line range

The FIND command searches a BASIC program for a string and displays the lines where it occurs. Because BASIC lines are tokenized it may be necessary to enclose the search string in quotes to find it. The whole program is searched unless a line range is given. The line range has the same format as the list command.

LIST A PROGRAM FROM THE DISK

Syntax: FLIST "program filename"

This command will list a BASIC program on the disk directly to the screen without affecting the contents in the memory. WARNING: DO NOT use the keyprint function to try and dump the screen to the printer while this command is executing.

DISPLAY WHERE AN ERROR OCCURED.

Syntax: HELP

The HELP command will display the BASIC line that caused the BASIC program to stop and highlight where in the line the problem occured.

CONVERT HEX AND DECIMAL NUMBERS

Syntax: HEX $hex number
HEX decimal number

The HEX command will convert HEX to decimal and decimal to HEX. This will be very useful in figuring POKE, PEEK, and SYS address. If the number input is preceded by a dollar symbol then the number is taken to be HEX and the decimal value for it is printed. If a decimal number is entered then the HEX value for it is returned. The range for conversion is 0 to 65535 or $0000 to $FFFF.

KILL BASIC-AID

Syntax: KILL

Basic-Aid can be disabled with the KILL command. This restores the IRQ vector and CHRGET routine in zero-page. Basic-Aid and be reenable with a SYS to the start of the Basic-Aid machine code, for the RAM version for 32K PETs this address is 7*4096.

PUT THE PET INTO LOWER CASE

Syntax: LOWER

This command puts the PET into lower case mode. This is the same as a POKE 59468,14.

MERGE A PROGRAM WITH ONE IN MEMORY

Syntax: MERGE "program filename"

This command will merge a BASIC program on the disk with one in memory. The merging will be just like the program was typed in from the keyboard so lines are merged between ones in memory if necessary and duplicate lines in memory are replaced with the merged lines. The program is listed as it is merged.

TURN OFF REPEAT AND SCROLL

Syntax: OFF

This command restores the PETs normal IRQ vector. This will cancel repeat keys (except on FAT 40s and 8032s), scrolling, and keyprint.

PACK A PROGRAM

Syntax: PACK

This command will remove remarks and waste spaces in a BASIC program. Note: don't branch in a BASIC program to deleted remarks. This command is fooled easily so keep a copy of the original in case the packing does not function properly.

READING A SEQUENTIAL FILE

Syntax: READ "seq filename"

The READ command will read a sequential file from the disk and print it to the screen. This command can be very handy for viewing data created by programs. WARNING: DO NOT use the keyprint function to try and dump the screen to the printer while this command is executing.

RENUMBER A BASIC PROGRAM

Syntax: RENUMBER
RENUMBER start line number
RENUMBER start line number, increment

This command will renumber a BASIC program correcting all GOTOs and GOSUBs in the program. The program is renumbered starting at 100 and with an increment of 10. A starting line number can be input other than 100 and an increment other than 10 can be input.

ENABLE REPEAT KEYS

Syntax: REPEAT
SCROLL (FAT 40s & 8032s)

This command will enable repeat keys, scrolling, and keyprint. Repeat keys are set automatically when Basic-aid is first called and automatically cancelled each time a program is loaded.

GIVE THE SIZE OF A PROGRAM

Syntax: SIZE
SIZE "program filename"

The SIZE command will give the size of a BASIC program in memory or any program on disk. The size of a program in memory is found by subtracting the end of the program location from the start of the program location. The size of a program on disk is found by counting the bytes in the file. The size is given in decimal and HEX.

SPOOL A FILE FROM DISK TO PRINTER

Syntax: SPOOL "sequential filename."
SPOOL

This command will send a file directly from the disk to the printer. The PET can then do other things, such as

Continued on page 28

Snow 'em with your VIC!

NEW

Simplify your printer set-up with

SMART ASCII $59.95

At last! A simple, convenient, low-cost printer interface. **It's ASCII:** connects the VIC or '64 to your favorite parallel printer (Epson, Microline, Smith-Corona TP-1, etc.). **It's SMART:** *translates* unprintable cursor commands and control characters for more readable LISTings.

Converts user port into parallel port with Centronics protocol, addressable as Device 4 or 5. **Three print modes:** CBM ASCII (all CAPS for LISTing); true ASCII (UPPER/lower case for text); and TRANSLATE (prints (CLR), (RED), (RVS), etc.). For **any size VIC** or the **'64**. Complete with printer cable and instructions.

UN-WORD PROCESSOR 2 . . . $19.95

The improved UN-WORD retains the practicality and economy of the original. Easy-to-use text entry and screen editing. Use with **any size VIC** (5K to 32K). Supports VIC printers, RS-232 printers, and now **parallel printers***, too. Handy user Menu selects: single- or double-space, form feed, print width, number of copies. Supports printer control codes. With complete documentation.

*Parallel printers require an interface. See SMART ASCII.

BANNER/HEADLINER $14.95

Make GIANT banners on your printer. Prints large characters across the page or sideways down the paper roll. . . how about a 10-ft. long "Welcome Home!". VIC or RS-232 printers.

VIC-PICs . . IMPROVED! . . $19.95

Now with **hi-rez draw routine** for your joystick *PLUS* hi-rez **dump to VIC printer.** Features 19 fascinating hi-rez *digitized* pictures. Capture *your* creativity, or ours, on paper. Amazing fun!

GRAFIX DESIGNER $14.95

Design *your own* graphic characters! Recall, erase, edit, copy, rotate . . . save to tape or disk for use in your own programs. Simple to use. Includes examples and demo routines.

GRAFIX MENAGERIE $14.95

Three-program set shows off VIC graphics potential for art, science, music, business . . . learn by seeing and doing. Contains BASIC **plotting routines** you may extract and use.

TERMINAL-40 . . . . . $29.95

Join the world of telecommunications in style: **40-character lines** and **smooth scrolling** text for easy reading! All software — no expensive hardware to buy. **4K** (or larger) **Receive Buffer** with optional dump to VIC printer. Function key access to frequently-used modes. Fully programmable Baud, Duplex, Parity, Wordsize, Stopbit, and Linefeed; supports control characters. Requires VIC-20, 8K (or larger) memory expansion and suitable modem. With 24 p. manual and Bulletin Board directory.

VIC-20 is a trademark of Commodore Business Machines

Introducing ***SOFTWARE FOR THE NEW***
COMMODORE 64

'64 TERMINAL ($29.95). Same impressive features as TERMINAL-40: **smooth-scrolling,** 40-character lines, VIC printer dump, etc. **GIANT 24K Receive Buffer.** No memory expansion required; requires '64 and modem.

'64 GRAFIX SAMPLER ($19.95). Indulge in the graphics splendor of the '64. Interact with demos of techniques such as **plotting** of points, lines and 3-D objects; **drawing** in the hi-rez mode (joystick control); animating **sprites;** plus assorted graphics displays. Routines may be extracted for use in your own programs.

'64 PANORAMA ($19.95). Explore picture graphics on the amazing '64! Nineteen fascinating *digitized* pictures PLUS **hi-rez draw routine** for your joystick AND hi-rez **dump to VIC printer**. Capture our pics or *your* creativity on paper.

'64 BANNER/HEADLINER ($19.95). Make **GIANT banners and posters** with your '64 and printer. Supports VIC printers, RS-232 printers (requires interface), and parallel printers (requires Smart Ascii).

ORDER DESK
Open 9 am - 4 pm
(816) 254-9600
VISA/Mastercard add 3%
C.O.D. add $3.50

MAIL ORDER: Add $1.25 shipping and handling. Send money order for fastest delivery. VISA/Mastercard send #and exp. date (3% added). Missouri residents include 4.6% sales tax. Foreign orders payable U.S.$, U.S. Bank ONLY; add $5 shipping/handling.

All programs on high quality digital cassette tape.

Write for free brochure. Dealer inquiries invited.

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

Continued from page 26

editing a program or running a program (but with no access to the IEEE bus). The command is started with the command and a filename of a sequential file name. Basic-Aid opens the file and listens. The printer then gets off the IEEE bus which allows the disk to talk directly to the printer. When the printer stops printing enter Spool with no file name to unlisten the printer, untalk the disk, and close the file.

Use the spool command to list a long program while you use the PET for something else. Create a file with
OPEN8,8,8,"0:TEMP,S,W":CMD8:LIST
PRINT#8:CLOSE8
Then spool this to the printer.

FIND THE LOAD ADDRESS

Syntax: START "program filename"

This command will give the load address of a program on the disk. The load address is found by reading the first two bytes of the file which is the address where the program is loaded. The load address is given in decimal and HEX. This command can be used to find where machine language programs load.

TRACE A PROGRAM EXECUTATION

Syntax: TRACE speed
TRACE

The TRACE command enables or disables the tracing of a BASIC program. Tracing is enabled with the command and a number and disabled with the command alone. The number input controls the speed of the tracing. The number can be from 1 to 127 with 1 being the fastest and 127 the slowest. Tracing takes place in a window in the upper right of the screen with the lost nine lines traced and the current line that is executing. The line number and what is executing in the line are listed.

RECOVER A PROGRAM AFTER A NEW

Syntax: UN-NEW

If after a New command is entered it is discovered that a program has not been saved it can be recovered with this command.

PUT THE PET INTO UPPER CASE

Syntax: UPPER

This command puts the PET into upper case mode. This is the same as a POKE 59468,12

DOS SUPPORT

Syntax: ►
►disk command
►$0
/program name
↑program name

The DOS support commands are supported. The at sign and greater than (@, ►) symbol are used to read the error channel, send commands, and display the disk directory. The symbol alone will read the error channel and print it to the screen. The symbol followed by a disk command will send that command to the disk. The symbol followed by the dollar symbol will display the directory to the screen. WARNING: DO NOT use the keyprint function to try and dump the screen to the printer while this command is executing. The slash (/) will load a program from the disk. Repeat keys are not disabled by this load. The uparrow (↑) will load and execute a program from disk.

ESCAPE QUOTE MODE

The stop key on graphics and the escape on business keyboards can be pressed to get out of insert mode and quote mode. This function will only work when repeat keys are enabled. (This is a normal feature on the 8032.)

KEYPRINT

This function allows the screen to be printed to the printer with the press of one key. This is the same as the CRT command except that it can occur in a program. On graphics keyboards use the shifted backslash and on business use the shifted escape. This function is available only when repeat keys are enabled. This can be very useful for printing instructions of a program that will then only print to the screen.

SCROLL A PROGRAM

The up and down cursor control keys can be used to scroll through a BASIC program. When the cursor is at the bottom of the screen and a cursor down is pressed, the next line will be printed and when the cursor is at the top of the screen and a cursor up is pressed, the previous line will be listed. The cursor must be in the first two columns for scrolling to take place. This function is only available when repeat keys are enabled.

NOTE: The commands which print to the screen (Change, Dump, Find, Flist, Merge, Read, Trace, and Directory (►$)) can be paused, held, or stopped. Pause by holding the shift key down and stop with the stop key. To hold the display use the space bar on graphics keyboards and 6 on business keyboards. To continue the display use the ◄ key on graphics keyboards and 9 on business keyboards.

ASSEMBLER-AID COMMAND LIST

BASIC-AID

Syntax: BASIC-AID

This command will disable these extra commands and initialize Basic-Aid only.

COLD

Syntax: COLD

The COLD command will do a software reset of the PET. This reset is like a power-on restart.

FORMATTED PRINT

Syntax: FORMAT (line range)

The FORMAT command is used to print the text file in tabbed format like the assembler. For this function to work correctly you must type mnemonics in column two, or one space from labels. The line range is the same as for a LIST. The same keys as for Basic-Aid can be used for to hold, pause, and stop the listing.

GET

Syntax: GET "file name"(,line #)

The GET command loads assembler text files from disk. This command is used to load source files into the editor and append to files already in memory. The file is input beginning at the line number given. GET starts numbering at 1000 by 10.

PUT

Syntax: PUT "dr:file name;;
(,line ;;="; range)
CPUT "dr:file name;;
(,line ;;="; range)

The PUT command outputs source

Continued on page 67

AARDVARK — THE ADVENTURE PLACE

ADVENTURES FOR OSI, TRS-80, TRS-80 COLOR, SINCLAIR, PET, VIC-20

ADVENTURES — Adventures are a unique form of computer game. They let you spend 30 to 70 hours exploring and conquering a world you have never seen before. There is little or no luck in Adventuring. The rewards are for creative thinking, courage, and wise gambling — not fast reflexes.

In Adventuring, the computer speaks and listens to plain English. No prior knowledge of computers, special controls, or games is required so everyone enjoys them—even people who do not like computers.

Except for Quest, itself unique among Adventure games, Adventures are non-graphic. Adventures are more like a novel than a comic book or arcade game. It is like reading a particular exciting book where you are the main character.

All of the Adventures in this ad are in Basic. They are full featured, fully plotted adventures that will take a minimum of thirty hours (in several sittings) to play.

Adventuring requires 16k on Sinclair, TRS-80, and TRS-80 Color. They require 8k on OSI and 13k on VIC-20. Sinclair requires extended BASIC.

TREK ADVENTURE by Bob Retelle — This one takes place aboard a familiar starship and is a must for trekkies. The problem is a familiar one — The ship is in a "decaying orbit" (the Captain never could learn to park!) and the engines are out (You would think that in all those years, they would have learned to build some that didn't die once a week). Your options are to start the engine, save the ship, get off the ship, or die. Good Luck.

Authors note to players — I wrote this one with a concordance in hand. It is very accurate — and a lot of fun. It was nice to wander around the ship instead of watching it on T.V.

CIRCLE WORLD by Bob Anderson — The Alien culture has built a huge world in the shape of a ring circling their sun. They left behind some strange creatures and a lot of advanced technology. Unfortunately, the world is headed for destruction and it is your job to save it before it plunges into the sun!

Editors note to players — In keeping with the large scale of Circle World, the author wrote a very large adventure. It has a lot of rooms and a lot of objects in them. It is a very convoluted, very complex adventure. One of our largest. Not available on OSI.

HAUNTED HOUSE by Bob Anderson — This one is for the kids. The house has ghosts, goblins, vampires and treasures — and problems designed for the 8 to 13 year old. This is a real adventure and does require some thinking and problem solving — but only for kids.

Authors note to players—This one was fun to write. The vocabulary and characters were designed for younger players and lots of things happen when they give the computer commands. This one teaches logical thought, mapping skills, and creativity while keeping their interest.

DERELICT by Rodger Olsen and Bob Anderson — For Wealth and Glory, you have to ransack a thousand year old space ship. You'll have to learn to speak their language and operate the machinery they left behind. The hardest problem of all is to live through it.

Authors note to players — This adventure is the new winner in the "Toughest Adventure at Aardvark Sweepstakes". Our most difficult problem in writing the adventure was to keep it logical and realistic. There are no irrational traps and sudden senseless deaths in Derelict. This ship was designed to be perfectly safe for its' builders. It just happens to be deadly to alien invaders like you.

NUCLEAR SUB by Bob Retelle — You start at the bottom of the ocean in a wrecked Nuclear Sub. There is literally no way to go but up. Save the ship, raise her, or get out of her before she blows or start WWIII.

Editors note to players—This was actually plotted by Rodger Olsen, Bob Retelle, and someone you don't know — Three of the nastiest minds in adventure writing. It is devious, wicked, and kills you often. The TRS-80 Color version has nice sound and special effects.

EARTHQUAKE by Bob Anderson and Rodger Olsen — A second kids adventure. You are trapped in a shopping center during an earthquake. There is a way out, but you need help. To save yourself, you have to be a hero and save others first.

Authors note to players — This one feels good. Not only is it designed for the younger set (see note on Haunted House), but it also plays nicely. Instead of killing, you have to save lives to win this one. The player must help others first if he/she is to survive — I like that.

PYRAMID by Rodger Olsen — This is one of our toughest Adventures. Average time through the Pyramid is 50 to 70 hours. The old boys who built this Pyramid did not mean for it to be ransacked by people like you.

Authors note to players — This is a very entertaining and very tough adventure. I left clues everywhere but came up with some ingenous problems. This one has captivated people so much that I get calls daily from as far away as New Zealand and France from bleary eyed people who are stuck in the Pyramid and desperate for more clues.

QUEST by Bob Retelle and Rodger Olsen — THIS IS DIFFERENT FROM ALL THE OTHER GAMES OF ADVENTURE!!!! It is played on a computer generated map of Alesia. You lead a small band of adventurers on a mission to conquer the Citadel of Moorlock. You have to build an army and then arm and feed them by combat, bargaining, exploration of ruins and temples, and outright banditry. The game takes 2 to 5 hours to play and is different each time. The TRS-80 Color version has nice visual effects and sound. Not available on OSI. This is the most popular game we have ever published.

MARS by Rodger Olsen — Your ship crashed on the Red Planet and you have to get home. You will have to explore a Martian city, repair your ship and deal with possibly hostile aliens to get home again.

Authors note to players — This is highly recommended as a first adventure. It is in no way simple—playing time normally runs from 30 to 50 hours — but it is constructed in a more "open" manner to let you try out adventuring and get used to the game before you hit the really tough problems.

ADVENTURE WRITING/DEATHSHIP by Rodger Olsen — This is a data sheet showing how we do it. It is about 14 pages of detailed instructions how to write your own adventures. It contains the entire text of Deathship. Data sheet - $3.95. NOTE: Owners of OSI, TRS-80, TRS-80 Color, and Vic 20 computers can also get Deathship on tape for an additional $5.00.

PRICE AND AVAILABILITY:

All adventures are $14.95 on tape except Earthquake and Haunted House which are $9.95. Disk versions are available on OSI and TRS-80 Color for $2.00 additional.

Please specify system on all orders

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

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

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

TRS-80 COLOR **SINCLAIR** **OSI** **VIC-20**

NEW WORD PROCESSOR FOR VIC 20/CMDR 64—$19.95 (tape) • $24.95 (disk)

Programing Time

by Hal Bredbenner

INTRODUCTION

One of the most useful features of Commodore computers is the Real Time Clock. This article describes some short subroutines that will simplify the use of the clock in your programs. Also included is a test program that will demonstrate the use of the subroutines. Hopefully the test program will spark some ideas of your own about how to use the clock. All Commodore systems include the Real Time Clock feature and the routines given will work on any one of them.

Years ago, as an early entry in the microprocessor market, Commodore introduced the PET 2001. At the time the PET was a very good buy because of the many features in one small package. 8K of RAM, an integral cassette deck, CRT with graphics, and MICROSOFT BASIC were hard to pass up at only $799.00. I suppose that because it had so many advantages over the competitors offerings some of its more useful functions were not well advertised. One such indispensible function that was on even the earliest models was a Real Time Clock (RTC). The RTC has been incorporated into the design of all Commodore systems since then. For some reason the RTC, which is directly setable and readable from BASIC, has not been used in many of the available software packages. Perhaps this is because most hardware vendors only supply a RTC as an option or costly addition.

UNDERSTANDING TI$

Before using the RTC, which is always available as the string variable TI$, you must understand the format in which the time is presented. The TI$ is always six characters in length. The first two are the hours, the middle two characters are the minutes, and the last two are the seconds. In addition to this HHMMSS format the RTC is also a 24 hour clock. A 24 hour clock is the format used in the military and most communications fields. To convert standard time to 24 hour clock time simply add 12 hours to the standard time if it is after 12:00 noon. For example 2:00 p.m. standard time is 14:00 24 hour clock time.

MAKING TIME CALCULATIONS

Whether you use the RTC for timed data logging, device scheduling, or game timing you will eventually want to perform some mathematical operations on the time. Perhaps the total time between events, time left in a cycle, or average duration of a cycle must be calculated. Since the RTC is available as a string variable some manipulation must take place before any time calculations can be made. The most common way of reducing this data to a usable format is to convert the HHMMSS string to its equivalent number of seconds. Any computations can then be made to this numeric quantity and then a conversion made back to the HHMMSS format. Using the string manipulation and numeric-to-string conversion statements of BASIC these conversions are fast and simple.

TIME CONVERSION SUBROUTINES

The subroutine at line 1000 of LISTING ONE converts the HHMMSS format to a total number of seconds. Enter the subroutine with X$ set to the HHMMSS time needed to convert. Upon returning from the routine X is equal to the total number of seconds in the time given. When the subroutine at line 2000 is called X should be equal to the number of seconds to convert. The correct HHMMSS format for the number of seconds is returned in X$. In line 2060 a check is made to see if complete days, groups of 84600 seconds, are included in the X value. If they are they are included in the returned data by placing the variable D equal to the number of complete days.

FURTHER FORMATTING SUBROUTINES

Frequently it is necessary to not only perform calculations using the data in the RTC but also to display the time on the CRT. The time in HHMMSS format is a little crude for direct display so the subroutine at line 3000 allows a more readable format of the time data to be generated. Two colons are inserted into the HHMMSS string assigned to X$ so that the string ends up like HH:MM:SS. I have found this subroutine saves memory rather than having to do the string manipulations each time in the PRINT statement. The final subroutine of LISTING ONE at line 4000 is just the reverse of the previous one. It removes the colons from the HH:MM:SS format and returns a HHMMSS format that can be used for further processing.

A DEMONSTRATION PROGRAM

LISTING TWO is of a short program that uses the four subroutines just described. The program first prompts for an INPUT of TI$, the RTC string variable. The string input should be six characters in length and in the HHMMSS format. After printing the time headers (lines 140-172) a loop is entered. This loop (lines 190-280) checks for a key depression and constantly updates the screen display with

the real and calculated time values. When a key is depressed the program jumps to line 300 where the times shown are operated on mathmatically. Note in line 400 that any value (in this case AT) should be rounded, truncated, or otherwise made into a whole integer form for use in these PTC subroutines.

The RTC routines given can be used in a variety of programs. Program execution time, printing time, average user response time and countdown timing for games are but a few ways to use the clock. On Commodore systems the RTC is available whether it is used or not, so why not use it to its fullest?

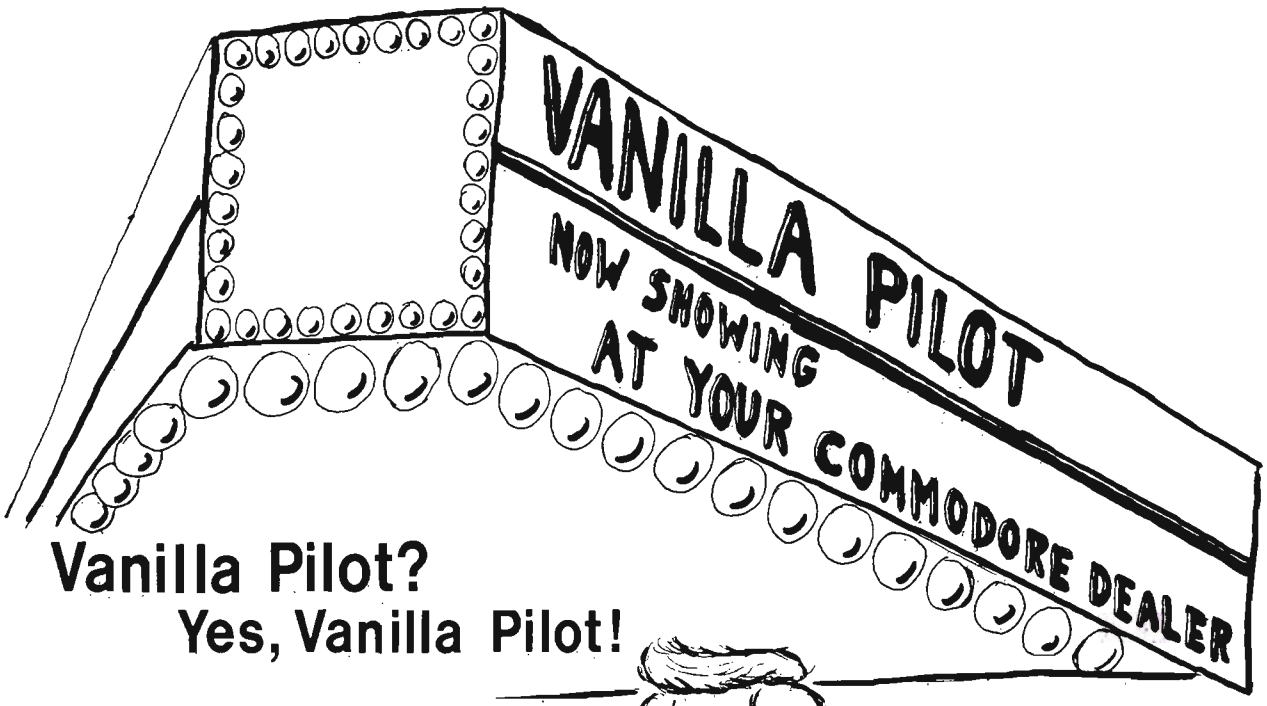
LISTING ONE

```
1000 REM** HHMMSS TO TOTAL SECS ****
1010 REM   ENTER WITH X$ = HHMMSS
1020 REM   RETURNS WITH X = TOTAL SECS
1030 REM
1040 H$=LEFT$(X$,2):M$=MID$(X$,3,2):S$=RIGHT$(X$,2)
1050 H=VAL(H$):M=VAL(M$):S=VAL(S$)
1060 X=(H*3600)+(M*60)+S
1070 RETURN
1080 REM
2000 REM** SECONDS TO HHMMSS ********
2010 REM   ENTER WITH X = # OF SECONDS
2020 REM   RETURNS WITH X$ = HHMMSS
2030 REM   ALSO RETURNS D = # OF DAYS
2040 REM
2050 D=0
2060 IFX>84600THENX=X-84600:D=D+1:GOTO2060
2070 H=INT(X/3600):M=INT((X-(H*3600))/60)
2080 S=X-(H*3600)-(M*60)
2090 H$=RIGHT$(STR$(H+100),2):M$=RIGHT$(STR$(M+100),2)
     :S$=RIGHT$(STR$(S+100),2)
2100 X$=RIGHT$(H$,2)+RIGHT$(M$,2)+RIGHT$(S$,2)
2110 RETURN
2120 REM
3000 REM** HHMMSS TO HH:MM:SS ********
3010 REM   ENTER WITH X$
3020 REM   RETURNS X$
3030 REM
3040 S$=LEFT$(X$,2)+":"+MID$(X$,3,2)+":"+RIGHT$(X$,2)
3050 X$=S$:RETURN
3060 REM
4000 REM** HH:MM:SS TO HHMMSS ********
4010 REM   ENTER WITH X$
4020 REM   RETURNS X$
4030 REM
4040 S$=LEFT$(X$,2)+MID$(X$,4,2)+RIGHT$(X$,2)
4050 X$=S$:RETURN
READY.
```

LISTING TWO

```
100 REM
110 REM  TIME TEST PROGRAM
120 REM
130 PRINT"■■■ENTER CURRENT TIME (HHMMSS)";:INPUTTI$
140 PRINT"■■■■PRESS ANY KEY...
150 PRINT"■■  ■CURRENT■    ■TIME SINCE■    ■TIME■
160 PRINT"■TIME OF DAY■   ■LAST KEY■    ■PRESSED■
170 PRINT"■■■■■AVG TIME BETWEEN■
```

```
171 PRINT"    ■EPRESSIONS■
172 PRINT"■■■■■■PRESS 'S' TO END
180 LK$=TI$:ET$="000000":AT$="000000"
190 GETS$:IFS$="S"THENEND
200 IFS$<>""THEN300
210 PRINT"■■■■■■■■■■■■
220 X$=TI$:GOSUB3000
230 PRINT" "X$;
240 X$=ET$:GOSUB3000
250 PRINT"      "X$;
260 X$=LK$:GOSUB3000
270 PRINT"     "X$
271 PRINT"■■■■":X$=AT$:GOSUB3000:PRINT"     "X$
280 GOTO190
290 REM
300 X$=TI$:GOSUB1000
310 A1=X
320 X$=LK$:GOSUB1000
330 B1=X
340 X$=TI$:GOSUB3000:LK$=TI$
350 X=A1-B1
360 GOSUB2000
370 ET$=X$
380 GOSUB1000
390 AV=AV+X:PC=PC+1
400 AT=INT(AV/PC)
 READY.
```



Vanilla Pilot?
Yes, Vanilla Pilot!

What is Vanilla Pilot?

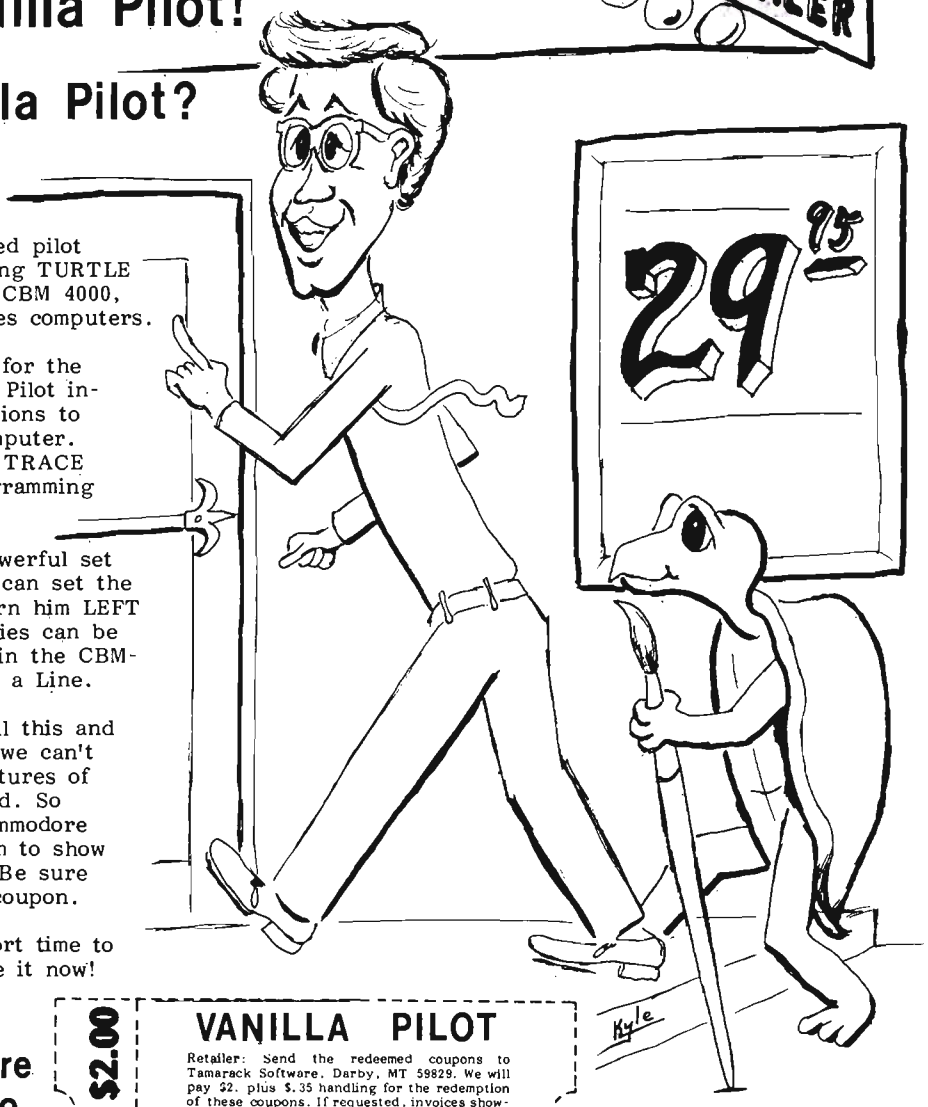
Vanilla Pilot is a full-featured pilot language interpreter including TURTLE GRAPHICS for the PET or CBM 4000, 8000, 9000 and CBM-64 series computers.

At last! A Pilot interpreter for the Commodore computers. This Pilot includes some powerful extensions to the screen editor of the computer. Things like FIND/CHANGE, TRACE and DUMP enhance the programming environment.

The TURTLE has a very powerful set of graphics commands. You can set the Turtle's DIRECTION and turn him LEFT or RIGHT. The pen he carries can be set to any of the 16 colors in the CBM-64. He can DRAW or ERASE a Line.

What else? Vanilla Pilot is all this and much, much more. In fact, we can't tell you about all of the features of the language in this small ad. So rush down to your local Commodore computer dealer and ask him to show you Vanilla Pilot in action. Be sure to take the $2.00 discount coupon.

Hurry, you have only a short time to redeem your coupon. So use it now!



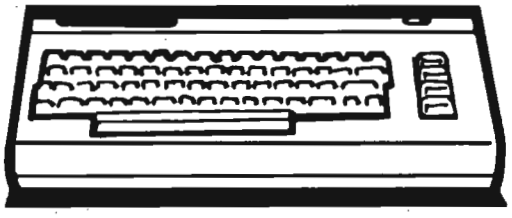
Tamarack Software
Darby, MT. 59829

SAVE $2.00

VANILLA PILOT

Retailer: Send the redeemed coupons to Tamarack Software. Darby, MT 59829. We will pay $2. plus $.35 handling for the redemption of these coupons. If requested, invoices showing sufficient purchase of Vanilla Pilot must be submitted. Coupons submitted to us more than 30 days after the expiration date will not be honored.

Expires April 15, 1983.



64

A Sprite Editor for the Commodore 64

by Garry Kiziak

One of the most exciting features of the new COMMODORE 64 computer is its ability to display high resolution graphics in up to 16 different colours. These graphics can be achieved in several different ways. In bit-mapped mode a resolution of 320 pixels by 200 pixels can be achieved. RAM based character sets can provide high resolution capabilities on the normal text screen. Finally sprite graphics provide bit-mapped resolution on the normal text screen with special features not available through RAM character sets—notice that I said the normal text screen, not just the high resolution screen.

Briefly, sprite graphics can be compared to the high resolution shapes of the APPLE or the player/missile graphics of the ATARI. However, the implementation of the COMMODORE 64 is without question superior. Sprites can be multi-coloured. They have different priority levels—enabling one sprite to be displayed in front of another (without requiring special filtering techniques). Collision detection between different sprites is handled automatically. They can be expanded, contracted and moved about easily. The most important feature is that this can all be accomplished easily within BASIC. This means that arcade quality graphics will become available to the BASIC programmer. The potential for educational programs is limited only by your imagination.

This summer I had the opportunity to work for COMMODORE CANADA, under the direction of Frank Winter, on a project that involved the use of COMMODORE 64 computers. An offshoot

```
10 REM
15 REM  SPRITE EDITOR BOOT
20 REM
25 PRINT "■":POKE 53280,14:POKE 53281,6
30 PRINT"■■■■■■■■■■■■■■■■■■■■■■■■■■■■■LOAD
ING ...":gosub 100
40 PRINT "■■LOAD"CHR$(34)"SCROLL.DATA"CH
R$(34)",8,1"
41 PRINT "■■■■NEW"
42 PRINT "■■LOAD"CHR$(34)"SPRITE EDITOR"
CHR$(34)",8"
50 POKE 198,10
55 DATA 19,13,13,13,32,32,82,85,78,13
60 FOR I=1 TO 10:READ X:POKE 630+I,X:NEX
T
70 POKE 44,64:POKE 16*1024,0:POKE 56,126
:CLR:NEW
100 PRINT"
110 PRINT"
120 PRINT"
130 PRINT"
140 PRINT"
150 PRINT"
160 PRINT"
170 PRINT"
180 RETURN
EADY.
```

of this project was the development of two programs that should simplify even further the use of high resolution graphics. The first program, which is presented in this article, is a SPRITE EDITOR—to simplify the creation of sprites. The second program, to be presented next month, is a CHARACTER EDITOR—to simplify the creation of RAM-based character sets.

Some Details About Sprites

A sprite is a high resolution shape that is 24 pixels wide and 21 pixels tall. Figure 1. shows an example. The pixels that are turned on (indicated in black) determine the shape that is displayed on the screen. This shape can be displayed in 16 different colours, but usually only one colour at a time. (Multi-coloured sprites are possible through specially arranged bit patterns and the use of the colour registers—see your manuals for details.)

To display a sprite, the computer must know which pixels to turn on and which pixels to turn off. This information is stored in 64 consecutive bytes of memory called a page or more accurately a sprite page. (Do not confuse this with the usual page concept associated with the 6502 or 6510 which is 256 bytes long.)

Why 64 bytes? And how is this data organized? Well, if you look at a single line of a sprite, say line 1 in Figure 1, there are 24 bits of information required—one bit for each pixel. This information can be stored in 3 bytes of memory if we match each pixel with a corresponding bit in the byte—0 means the pixel is turned off, 1 means the pixel is turned on.

For example, the first 8 pixels would correspond to the bit pattern (i.e. binary number) 00110000 which is 48 decimal. Similarly the next 8 pixels would correspond to 00001110 or 14 decimal while the last 8 pixels would correspond to 00000001 or 1 decimal. Thus the pixel information for the first line of the sprite in Figure 1. would be contained in the three numbers (bytes) 48, 14, and 1. Twenty-one such lines would require 21 x 3 or 63 bytes of storage. To simplify the organizational structure of sprite data, the COMMODORE 64 actually uses 64 bytes (one byte, the last one, is wasted).

Hopefully you can see that creating the data for a complete sprite manually in this way would require a substantial amount of work. Creating a series of sprites this way would be very time consuming and error prone. Finally, editing a sprite, that didn't quite look like you wanted it to, would require a very careful examination of the data that you had created previously and changing appropriate values, again a time consuming and error prone venture.

This Sprite Editor will take all the hard work out of creating sprites and in fact should even make it an enjoyable experience. The inclusion of sprites within your own programs should become a natural and common occurence, as it was intended to be.

The sprite pages, that we mentioned above, must occupy specific portions of the computer's memory. For example, the first page, called page 0, occupies the first 64 bytes of memory (locations 0-63). Page 1 occupies the next 64 bytes (locations 64-127), and so on. Two hundred and fifty-six sprite pages are possible with page 255 occupying locations 16320-16383.

Notice that up to 16K of memory can be devoted to sprite definitions—at least in theory. In reality many of these pages are used by the operating system and are not available for sprite usage. The actual breakdown is as follows.

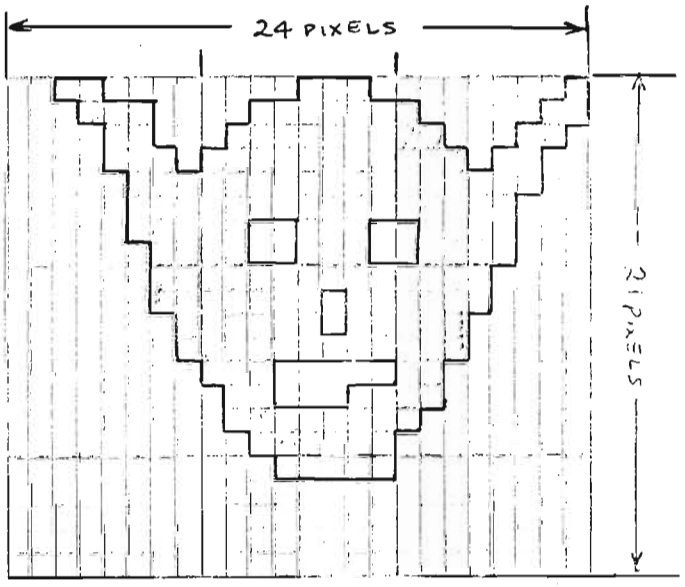


FIGURE 1

Sprite Pages	Memory Locations	Status	Reason
0 - 12	0 - 831	Not Avail.	Zero page, operating system pointers, etc.
13 - 15	832 - 1023	Available	Cassette Buffer.
16 - 31	1024 - 2047	Not Avail.	Screen Memory and sprite pointers.
32 - 63	2048 - 4095	Available	See note 3) below.
64 - 127	4096 - 8191	Not Avail.	I don't know why!!
128 - 255	8192 - 16383	Available	See note 3) below.

Notes:

1) Why sprite pages 64-127 are not available is not clear to me at this time. I only know that they don't work. Putting a sprite definition into this area and then trying to display it using the appropriate pointers does not produce the shape that was defined.

2) Although 256 sprite definitions are theoretically possible, only 163 are actually usable. Also only 8 sprites can be displayed at any given time (at least from BASIC).

3) A BASIC program is normally stored starting at locations 2048. Thus one might assume that pages 32-63 (and possibly some of 128-255, depending on the length of the program) are not available. As we shall shortly see, it is easy to move a BASIC program upwards in memory to start at any loation that we choose. With this in mind, we can consider sprite pages 32-63 and 118-255 as being available. The Sprite Editor itself will be moved beyond the 16K mark so that sprites can be created and edited on these pages.

Using the Sprite Editor

For the moment, I will assume that the Sprite Editor (Listings 1, 2, and 3) has been typed in, debugged, and is up and running.

You should see a 24 x 21 grid situated at the left side of the screen with a cursor flashing at the upper left (home position).

All the commands that allow you to create and edit a sprite on this grid are invoked by pressing a single key (or two keys simultaneously—the CTRL key and some other key).

For example, the '.' key (the period key) when pressed will plot a large dot at the position of the cursor on the edit grid—the cursor will also move one position to the right. In addition, the actual sprite will begin to take form to the right of the grid with the corresponding pixel being turned on. Press the '.' key again several times in succession. A line of dots should appear on the edit grid. As well, the corresponding pixels of the sprite should be turned on at the right.

The cursor can be moved around without turning on or off any pixels, using the normal cursor keys (up, down,

```
99 :
100 PRINT "■";:IF G=0 THEN DIM CO$(15)
110 DEF FNX(X)=X-INT(X/24)*24
120 DEF FNY(X)=X-INT(X/21)*21
130 V$="■■■■■■■■■■■■■■■■■■■■■■■■■"
140 DOT$=".......................":BL$=
"                                       "
142 CO$(0)="BLACK":CO$(1)="WHITE":CO$(2)
="RED  ":CO$(3)="CYAN ":CO$(4)="PURPL"
144 CO$(5)="GREEN":CO$(6)="BLUE ":CO$(7)
="YELOW":CO$(8)="ORNGE":CO$(9)="BROWN"
146 CO$(10)="LT RD":CO$(11)="GRAY1":CO$(
12)="GRAY2":CO$(13)="LT GN"
148 CO$(14)="LT BL":CO$(15)="GRAY3":MR=P
EEK(53276)
149 M0=PEEK(53285)AND15:M1=PEEK(53286)AN
D15:B=PEEK(53281)AND15:E=PEEK(53280)AND15
150 G=13*4096:CR$=CHR$(13):DE$=CHR$(20):
C=PEEK(G+39)AND15:SX=30:SY=150:X1=0:Y1=0
160 PA=200:SP=0:SC=1024+80+3:AD=32608
170 GOSUB 960
180 GOSUB 870
190 GOSUB 930
200 GOSUB 850
210 PX=0:PY=0
220 P=SC+PY*40+PX:Q=PEEK(P):R=Q
230 R=(NOTRAND128)OR(NOT128ANDR)
240 POKE P,R
250 FOR I=1 TO 30:GET A$:IF A$="" THEN N
EXT:GOTO 230
260 POKE P,Q
270 IF A$="■" THEN PX=FNX(PX+1):GOTO 220
280 IF A$="■" THEN PX=FNX(PX-1):GOTO 220
290 IF A$="■" THEN PY=FNY(PY+1):GOTO 220
300 IF A$="■" THEN PY=FNY(PY-1):GOTO 220
310 IF A$=DE$ THEN POKE P,PEEK(P) AND 12
7:GOTO 680
320 IF A$=" " THEN GOSUB 660:PX=FNX(PX+1
):GOTO 220
330 IF A$=CR$ THEN PX=0:PY=FNY(PY+1):GOT
O 220
340 IF A$="." THEN 740
350 IF A$="■" THEN GOSUB 820:GOTO 210
360 IF A$="■" THEN GOSUB 830:GOTO 210
370 IF A$="+" THEN 780
380 IF A$="-" THEN 800
390 IF A$="Q" THEN POKE G+21,0:PRINT "■"
;:END
400 IF A$=">" THEN C=(C+1)AND15:POKE G+3
9,C:GOSUB 3020
410 IF A$="■" THEN B=(B+1)AND15:POKE 532
81,B:GOSUB 3010
420 IF A$="■" THEN E=(E+1)AND15:POKE 532
80,E:GOSUB 3000
430 IF A$="■" THEN X1=1-X1:GOSUB 900:GOS
UB 870:GOTO 220
440 IF A$="■" THEN Y1=1-Y1:GOSUB 900:GOS
UB 870:GOTO 220
```

left, and right)—notice that when you reach the edge of the grid, the cursor wraps around to the opposite edge of the same horizontal/vertical line.

The space key can be used to erase any plotted dot under the cursor and turn the corresponding pixel of the sprite off. This is essentially how a sprite is created.

'.' turns a pixel on

space turns a pixel off

while the cursor keys move the cursor around without affecting any pixels. However, a number of other commands are included to make the entire editing process somewhat easier. These commands are summarized below.

Keystroke(s) Action

HOME—Moves the cursor to the upper left (home position) of the edit grid. Nothing else on the screen is affected.

CLR—Clears the entire edit grid (as well as the corresponding sprite).

DEL—Deletes any point to the left of the current cursor position.

RETURN—Moves the cursor to the beginning of the next line.

```
450 IF A$="■" THEN 580
460 IF A$="■" THEN 600
470 IF A$="▐" THEN 620
480 IF A$="▌" THEN 640
490 IF A$="■" THEN 1510
500 IF A$="S" THEN 1270
510 IF A$="■" THEN 1030
520 IF A$="■" THEN 1070
530 IF A$="■" THEN 1430
540 IF A$="■" THEN 210
550 IF A$="■" THEN GOSUB 1860:GOTO 170
555 IF A$="£" THEN 690
556 IF A$="M" THEN MR=255-MR:POKE 53276,
MR:GOSUB 3030
557 IF A$="0" THEN M0=(M0+1)AND15:POKE 5
3285,M0:GOSUB 3030
558 IF A$="1" THEN M1=(M1+1)AND15:POKE 5
3286,M1:GOSUB 3030
560 GOTO 220
570 POKE G+21,0:GOSUB 930:GOSUB 870:GOSU
B 850:GOTO 210
574 REM ******************************
575 REM *                            *
576 REM *  MOVE ENTIRE SPRITE UP,    *
577 REM *   DOWN, LEFT, OR RIGHT     *
578 REM *                            *
579 REM ******************************
580 J=PA*64:POKE 253,J-256*INT(J/256):PO
KE 254,J/256
```

Notice the similarities between the action of these keystrokes and their action when editing a normal BASIC program.

CTRL-P—Prompts you for the page number of the sprite that you wish to edit—only the pages available above are accepted. When the editor is first run, page 200 is displayed as the default value, but you can begin wherever you wish by using this command.

+ —Displays the next sprite for editing. In most applications, you will likely use more than one sprite. The definitions for these sprites will likely be stored one after the other in memory. When you finish working on one sprite page, use this command to display the next page so you can begin creating the next sprite.

– —Displays the previous page for editing.

Note: Moving from one page to another does not destroy the contents of either page.

CTRL-R—This reverses the sprite currently being edited; that is, each pixel that is on is turned off and each pixel that is off is turned on.

F_1—Moves the entire grid up one line. The data that was originally on the first line is now lost and the bottom line is blanked. You will likely use this command when you find that the sprite you are editing is not positioned exactly as you would like it.

F_3—Moves the entire grid down one line.

F_5—Moves the entire grid left one column.

F_7—Moves the entire grid right one column.

(The Pound Key)—This rotates the sprite 90 degrees. Notice that the rotated sprite will be slightly distorted. This is because the horizontal and vertical sides of a pixel are not the same length.

The rotate command poses one potential difficulty. It arises from the fact that we are rotating a shape that is not a perfect square (i.e. a 24 x 21 sprite). To get around this problem, when the pound key is first pressed,

```
590 SYS AD:GOSUB 850:GOTO220
600 J=PA*64+59:POKE 253,J-256*INT(J/256)
:POKE 254,J/256
610 SYS AD+42:GOSUB 850:GOTO220
620 J=PA*64:POKE 253,J-256*INT(J/256):PO
KE 254,J/256
630 SYS AD+88:GOSUB 850:GOTO220
640 J=PA*64:POKE 253,J-256*INT(J/256):PO
KE 254,J/256
650 SYS AD+118:GOSUB 850:GOTO220
655 REM ******************************
656 REM *                            *
657 REM * ERASE OR DELETE A POINT    *
658 REM *                            *
659 REM ******************************
660 POKE P,46:PP=PA*64+PY*3+INT(PX/8)
670 POKE PP,PEEK(PP) AND 255-2↑(7-(PX-IN
T(PX/8)*8)):RETURN
680 PX=FNX(PX-1):P=SC+PY*40+PX:GOSUB 660
:GOTO 220
685 REM ******************************
686 REM *                            *
687 REM * ROTATE SPRITE 90 DEGREES   *
688 REM *                            *
689 REM ******************************
690 HI=INT(PA/4):LO=PA*64-256*HI:POKE 25
1,LO:POKE 252,HI:SYS 32422
700 SYS 32443:SYS 32526:POKE 251,LO:POKE
 252,HI
710 GET A$:IF A$<>"£" AND A$<>CR$ THEN 7
10
720 IF A$="£" THEN 700
730 GOTO 220
735 REM ******************************
736 REM *                            *
737 REM *        PLOT A POINT        *
738 REM *                            *
739 REM ******************************
740 POKE P,81
750 PP=PA*64+PY*3+INT(PX/8)
760 POKE PP,PEEK(PP) OR 2↑(7-(PX-INT(PX/
8)*8))
770 PX=FNX(PX+1):GOTO 220
775 REM ******************************
776 REM *                            *
777 REM * NEXT OR PREVIOUS SPRITE    *
778 REM *                            *
779 REM ******************************
780 IF PA<15 OR (PA>31ANDPA<63) OR(PA>12
7ANDPA<255) THEN PA=PA+1:GOTO570
790 GOTO 220
800 IF(PA>13ANDPA<16) OR (PA>32ANDPA<64)
 OR(PA>128ANDPA<256) THEN PA=PA-1:GOTO570
810 GOTO 220
815 REM ******************************
816 REM *                            *
817 REM *        CLEAR SPRITE        *
818 REM *                            *
819 REM ******************************
```

the sprite is put into a 24 x 24 buffer with the bottom 3 lines initially all blank. This buffer is then rotated 90 degrees. The top 21 lines of this buffer are then displayed on the screen, the bottom 3 lines are not.

When you press the pound key to rotate the sprite, you will notice that the flashing cursor disappears. At this point, only two keystrokes are recognized. The pound key can be pressed again (or as often as you like). Each keypress will rotate the sprite, actually the buffer, another 90 degrees. So for example, pressing the pound key twice in succession will turn the sprite upside down.

The other acceptable keypress is the RETURN key. This causes the flashing cursor to reappear signifying that you are once again in EDIT mode. One other important thing happens however. Whatever is visible on the screen (i.e. the top 21 lines of the buffer) is stored in the sprite definition and whatever was in the last 3 lines of the buffer is lost forever.

This has the following consequences. Suppose you press the pound key to rotate a sprite and follow it with the RETURN key. Then you decide that you really wanted to turn it upside down, so you press the pound key once again. Chances are, depending on the size of your sprite, that you will have lost a portion of it; namely, whatever was in the bottom 3 lines of the buffer. The only way to avoid this is to make certain that you have rotated the sprite as many times as you require before you press the RETURN key.

More Commands

Keystrokes Action

CTRL-X—Expands/contracts the size of the sprite horizontally. This key acts as a toggle; that is, pressing it the first time expands (doubles) the size of the sprite. Pressing it a second time restores it to its normal size.

CTRL-Y—Expands/contracts the size of the sprite vertically—again it acts like a toggle.

CTRL-B—Each keypress here will change the background to a new colour. Actually it steps through the

```
820 FOR X=0 TO 63:POKE PA*64+X,0:NEXT:GO
SUB 850:RETURN
825 REM ******************************
826 REM *                            *
827 REM *      REVERSE SPRITE        *
828 REM *                            *
829 REM ******************************
830 FOR X=0 TO 63:Y=PEEK(PA*64+X):Y=(NOT
YANDZ55) OR (NOT255ANDY):POKE PA*64+X,Y
840 NEXT:GOSUB 850:RETURN
845 REM ******************************
846 REM *                            *
847 REM * DISPLAY SPRITE ON SCREEN   *
848 REM *                            *
849 REM ******************************
850 POKE 251,PA*64-INT(PA/4)*256:POKE 25
2,PA/4
860 SYS 32526:PRINT"■■":RETURN
865 REM ******************************
866 REM *                            *
867 REM *  INITIALIZE SPRITE DATA    *
868 REM *                            *
869 REM ******************************
870 POKE G+21,0:POKE 2040+SP,PA:POKE G+3
9,C
880 POKE G+SP*2,SX:POKE G+SP*2+1,SY:POKE
 G+23,Y1:POKE G+29,X1
890 POKE G+16,2↑SP:POKE G+21,2↑SP:RETURN
895 REM ******************************
896 REM *                            *
897 REM *  EXPAND/CONTRACT SPRITE    *
898 REM *                            *
899 REM ******************************
900 SX=17:IF X1=0 THEN SX=30
910 SY=138:IF Y1=0 THEN SY=150
920 RETURN
925 REM ******************************
926 REM *                            *
927 REM *  DISPLAY EDITING SCREEN    *
928 REM *                            *
929 REM ******************************
930 PRINT "■■■■"TAB(29)"|          |":PRIN
T TAB(29)"| PAGE     |"
940 PRINT TAB(29)"└─────────┘":PRINT "■■"
TAB(34)PA
950 RETURN
960 PRINT "■ ──────────────────────────
  ──────────"
970 PRINT " ■  12345678901234567890123
  ■SPRITE  editor■"
980 FOR X=1 TO 21
990 PRINT " ■"RIGHT$(" "+STR$(X),2)"■"DO
T$"■ ■":NEXT
1000 PRINT " ■
 "
1001 PRINT LEFT$(V$,18)TAB(30)"COLOURS":
PRINTTAB(30)"──────"
```

16 colors—black, white, red, cyan, purple, green, blue, yellow, orange, brown, light red, gray 1, gray 2, light green, light blue, and gray 3 so that pressing it 16 times in succession will restore the background to its original colour. The current background colour is always displayed near the bottom right of the screen.

This command should be useful if you have an application program that makes use of sprites on a background different from that used by the Sprite Editor. It allows you to view your sprite as it would appear in your program on the background of your choice.

CTRL-E—This has the same effect as CTRL-B only applied to the Border of Edge around the screen. The current border colour is also displayed at the right of the screen.

►—This allows you to change the colour of the sprite being edited—again it steps through the 16 colours and is displayed at the right of the screen.

M—This allows you to change to Multicolour sprite mode. Actually, it acts as a toggle, allowing you to change back and forth between Multicolour mode and Standard mode. You can tell when you are in Multicolour mode because the colours in colour registers 0 and 1 are also displayed at the bottom right of the screen.

0 & 1—These keystrokes allow you to change the colours in the colour registers, again stepping through the sixteen available colours. Pressing 0 changes register 0, while pressing 1 changes the colour in register 1.

CTRL-D—This command allows you to display a range of sprites for viewing only. You are prompted for the beginning page number of the range as well as the ending page number. The sprites are then displayed 8 at a time until the entire range is displayed.

CTRL-C—This command allows you to copy a range of sprites from one area to another. You are prompted for the source pages (starting page and ending page) and the target pages (start only).

```
1002 PRINT TAB(29)"BRDR:":GOSUB 3000
1004 PRINT TAB(29)"BKGD:":GOSUB 3010
1006 PRINT TAB(29)"SPRT:":GOSUB 3020:IF
MR=0 THEN RETURN
1008 GOSUB 3030
1010 RETURN
1015 REM ******************************
1016 REM *                            *
1017 REM *    PROMPT FOR NEW PAGE     *
1018 REM *                            *
1019 REM ******************************
1020 IF IN$="" THEN PA=AP:GOTO 180
1030 LI=10:COL=31:LE=3:MSG$="■PAGE■":GOS
UB 1700:AP=PA:PA=VAL(IN$)
1040 PRINTLEFT$(V$,LI)TAB(COL-1)"
  "
1050 IF IN$="" OR PA<13 OR (PA>15ANDPA<3
2)OR (PA>63ANDPA<128) OR PA>255 THEN PA=AP
1060 GOTO 180
1065 REM ******************************
1066 REM *                            *
1067 REM * DISPLAY RANGE OF SPRITES   *
1068 REM *                            *
1069 REM ******************************
1070 POKE G+21,0:POKE G+16,0:POKE G+23,0
:POKE G+29,0:GOSUB 1250
1080 LI=8:COL=6:LE=3:MSG$="■FROM PAGE:■"
1090 GOSUB 1700:GP=PG:PG=VAL(IN$):IFIN$=
"" THEN PRINT "❑":GOTO 170
1100 IF PG<0 OR PG>255 OR (PG=0 AND IN$<
>"0") THEN 1080
1110 LI=8:COL=22:MSG$="■TO PAGE:■"
1120 GOSUB 1700:PH=VAL(IN$):IF PH<PG OR
PH>255 THEN 1110
1130 SW=PG
1140 SUM=0:EN=SW+7:IF EN>PH THEN EN=PH:I
F SW>PH THEN 170
1150 GOSUB 1250:FOR I=SW TO EN:K=I-SW:M=
K:IF M>3 THEN M=M-4
1160 POKE 2040+K,I:SUM=SUM+2↑K
1170 POKE G+K*2,M*48+104:L=98:L1=10:IF K
>3 THEN L=155:L1=17
1180 POKE G+K*2+1,L:POKE G+39+K,1:POKE G
+21,SUM
1190 PRINT LEFT$(V$,L1)TAB(M*6+9)I
1200 NEXT
1210 PRINTLEFT$(V$,20)TAB(7)"
  "
1220 PRINTTAB(7)"PRESS ■SPACE BAR■ TO CO
NTINUE"
1230 GET A$:IF A$<>" "THEN 1230
1240 POKE G+21,0:PRINT "❑":SW=EN+1:GOTO
1140
1250 PRINT "❑
  "
1260 PRINT "          ■ DISPLAY SPRITE
S ":RETURN
```

You could require this command in the development stages of an application program. For example, you may have created your sprites on certain pages located right after your BASIC program. However, as your program grows beyond your expectations, it starts to overwrite your sprites (or vice versa). Thus you have to move the sprites to a higher location in memory.

This copy command can move your sprites higher in memory, lower in memory, or even to overlap the source range.

You might wish to use this command when creating sprites to simulate motion. One sprite will be exactly the same as the previous one except for some minor change. Displaying these sprites rapidly in succession would give the impression of motion. The creation of these sprites can be simplified by first creating one of the sprites, then copying it to the next page, and then making the necessary changes to it.

S—This command allows you to save a range of sprite definitions to disk. You are prompted for a filename, and for the starting page and end-

```
1265 REM ****************************
1266 REM *                          *
1267 REM *   SAVE SPRITES TO DISK   *
1268 REM *                          *
1269 REM ****************************
1270 POKE G+21,0:PRINT "■
                     "
1280 PRINT "              ■ SAVE SPRITE DAT
A "
1290 LI=8:COL=6:LE=3:MSG$="■FROM PAGE:■"
1300 GOSUB 1700:PG=VAL(IN$):IF IN$="" TH
EN PRINT "■":GOTO 170
1310 IF PG<13 OR (PG>15 AND PG<32) OR (P
G>63 AND PG<128) OR PG>255 THEN 1290
1320 LI=8:COL=22:MSG$="■TO PAGE:■":GOSUB
 1700:PH=VAL(IN$)
1330 IF PH<PG OR (PH>15 AND PH<32) OR (P
H>63 AND PH<128) OR PH>255 THEN 1320
1340 BEG=PG*64:EN=PH*64+63:HI=INT(BEG/25
6):LO=BEG-HI*256
1350 LI=10:COL=12:LE=16:MSG$="■FILENAME:
":GOSUB 1700:FL$="0:"+IN$+",PRG,WRITE"
1360 OPEN 1,8,15,"I0":GOSUB 1810
1370 OPEN 2,8,1,FL$:GOSUB 1810
1380 PRINT#2,CHR$(LO);CHR$(HI);
1390 FOR I=BEG TO EN:X=PEEK(I):PRINT#2,C
HR$(X);:NEXT
1400 CLOSE 2
1410 CLOSE 1
1420 PRINT "■":GOTO 170
```

ing page to be saved. You might want to use the display command first to verify that you are selecting the right pages. The sprite definitions are saved as a PGM file and can later be loaded in by your application program.

CTRL-L—This loads a previously saved range of sprites into memory. You are prompted for the filename and then the sprites are loaded into exactly the same location from which they were saved. You will want to use this to edit sprites that you previously created, to move them to a different location in memory, etc.

CTRL-V—This command allows you to view the sprite, currently being edited, moving about on a clear screen. This should give you some idea as to how it will look in your program.

While the sprite is moving about, several keypresses are still recognized. CTRL-B, CTRL-E, CTRL-X, CTRL-Y, and ► have the same effect as mentioned above. The '+' key will speed up the motion, the '–' key will slow it down. Pressing the SPACE key will stop the motion and pressing it again will restart it. Finally, the RETURN key will return you to the Editor so you can continue editing your sprites.

The final command that the Sprite Editor will accept is the 'Q' command—used to quit the Editor. If you should press this key accidently, you can restart the Editor simply by typing RUN. All your sprites will still be intact and you can continue editing as before.

Typing in the Sprite Editor

To get the Sprite Editor up and running, you will have to type in three separate programs.

1. The SPRITE BOOT program (Listing 1.)

The Sprite Editor itself must be relocated to start at location 16384 (i.e. above the last possible sprite page). The purpose of the SPRITE BOOT program is to change the pointers necessary to effect this relocation. It also loads in the actual Sprite Editor and a machine language utility used by the editor. It uses the 'Dynamic

```
1425 REM ******************************
1426 REM *                            *
1427 REM *  LOAD SPRITES FROM DISK    *
1428 REM *                            *
1429 REM ******************************
1430 POKE G+21,0:PRINT "□                _____
_______________"
1440 PRINT "                ■ LOAD SPRITE DAT
A "
1450 LI=10:COL=12:LE=16:MSG$="■FILENAME:
":GOSUB 1700:FL$="0:"+IN$
1460 IF IN$="" THEN PRINT "□":GOTO 170
1470 OPEN1,8,15,"I0"
1480 OPEN2,8,0,FL$:GOSUB 1810:CLOSE 2:CL
OSE 1
1490 LOAD FL$,8,1
1500 END
1504 REM ******************************
1505 REM *                            *
1506 REM *  COPY SPRITES TO ANOTHER   *
1507 REM *       AREA IN MEMORY       *
1508 REM *                            *
1509 REM ******************************
1510 POKE G+21,0:PRINT "□                _____
_______________"
1520 PRINT "                ■ COPY SPRITE DAT
A "
1530 PRINT LEFT$(V$,4)" _________________
__":PRINT " ** SOURCE PAGES **"
1540 LI=7:COL=4:LE=3:MSG$="■FROM PAGE:■"
1550 GOSUB 1700:PG=VAL(IN$):IF IN$="" TH
EN PRINT "□":GOTO 170
1560 IF PG<13 OR (PG>15 AND PG<32) OR (P
G>63 AND PG<128) OR PG>255 THEN 1540
1570 LI=9:COL=4:LE=3:MSG$="■..TO PAGE:■"
:GOSUB 1700:PH=VAL(IN$)
1580 IF PH<PG OR (PH>15 AND PH<32) OR (P
H>63 AND PH<128) OR PH>255 THEN 1570
1590 PRINT LEFT$(V$,4)TAB(21)"___________
__________":PRINT TAB(21)"** TARGET PAGES **"
1600 LI=7:COL=24:LE=3:MSG$="■FROM PAGE:■
":GOSUB 1700:PD=VAL(IN$)
1610 IF PD<13 OR (PD>15 AND PD<32) OR (P
D>63 AND PD<128) OR PD>255 THEN 1600
1620 PE=PD+PH-PG:IF PE>255 THEN PE=255
1630 PRINTLEFT$(V$,9)TAB(23)"..TO PAGE:"
PE
1640 IF PD>PG AND PD<=PH THEN 1670
1650 FOR I=PD TO PE:PRINT LEFT$(V$,12)TA
B(12)"COPYING PAGE"I
1660 FOR J=0 TO 63:POKE I*64+J,PEEK((PG+
I-PD)*64+J):NEXT:NEXT:PRINT "□":GOTO 170
1670 FOR I=PE TO PD STEP -1:PRINT LEFT$(
V$,12)TAB(12)"COPYING PAGE"I
1680 FOR J=0 TO 63:POKE I*64+J,PEEK((PG+
I-PD)*64+J):NEXT:NEXT:PRINT "□":GOTO 170
1690 END
```

Keyboard' technique (so often used with the PET) to load these programs, but it does it in a way that the usual messages are not displayed on the screen.

2. The SPRITE EDITOR program (Listing 2.)

This is the actual Sprite Editor. The listing includes many REM statements to set off the various routines and make it easier to make any modifications or additions to the program. These can be omitted if you wish to reduce the size of the program (it can almost be cut in half).

3. The M/L DATA program (Listing 3.)

The Sprite Editor requires several machine language routines for rotating a sprite, moving a sprite up/down, etc. This program contains that M/L program in data statements and when run will create a PGM file called SCROLL.DATA that the SPRITE BOOT program loads in along with the Sprite Editor.

You need only run the M/L DATA program once. The SCROLL.DATA program will then be on your disk. From then on, whenever you want to use the Sprite Editor, simply load and run the SPRITE BOOT program. It will do all the necessary housekeeping chores to get the editor up and running.

Saving a Sprite Table Along with Your Basic Program

The Sprite Editor creates a PGM file for the sprites that you create using a name that you specify. This file, says SPRITE TABLE, can be loaded into memory by your BASIC program by including a statement such as the following as the first line of your program.

```
10 IF FLAG = 0 THEN FLAG = 1 :
   LOAD "SPRITE TABLE",8,1
```

Note:

1) The ',1' at the end of the load command is necessary since it tells the computer to load the program without relocating it. Leave it out and you will overwrite the beginning of your BASIC program.

2. When a program is loaded from within another program, that program is automatically run (with all variables created up to that point preserved). When the sprites are loaded in line 10,

```
1695 REM ******************************
1696 REM *                            *
1697 REM *       INPUT ROUTINE        *
1698 REM *                            *
1699 REM ******************************
1700 Y9=2:IN$="":UC=0:UB$=LEFT$(BL$,LE):
GOSUB 1800:UB$=" ":UC=3
1710 UT=TI
1720 GET Z9$:IF Z9$="" THEN 1780
1730 IF Z9$=CR$ THEN Y9=2:GOSUB 1800:PRI
NT "■■ ":RETURN
1740 IF Z9$=DE$ THEN ON -(LEN(IN$)=0) GO
TO 1780:IN$=LEFT$(IN$,LEN(IN$)-1):GOTO1780
1750 IF (ASC(Z9$)AND127)<32 OR Z9$=CHR$(
34) THEN 1780
1760 IF LE=LEN(IN$) THEN 1780
1770 IN$=IN$ + Z9$
1780 GOSUB 1800:IF TI-UT<10 THEN 1720
1790 Y9=3-Y9:GOTO 1710
1800 PRINT LEFT$(V$,LI)TAB(COL-1)MID$(MS
G$,Y9)UB$IN$MID$("■ ■",Y9,UC)" ";:RETURN
1805 REM ******************************
1806 REM *                            *
1807 REM *  CHECK FOR DISK ERRORS     *
1808 REM *                            *
1809 REM ******************************
1810 INPUT#1,A$,B$,C$,D$
1820 IF VAL(A$)=0 THEN RETURN
1830 PRINT"■■■DISK ERROR:■ "B$
1840 CLOSE2
1850 END
1855 REM ******************************
1856 REM *                            *
1857 REM *  VIEW SPRITE IN MOTION     *
1858 REM *                            *
1859 REM ******************************
1860 POKE G+21,0:PRINT "■":POKE G+16,0
1870 X=INT(RND(1)*100)+155:Y=INT(RND(1)*
100)+75:DX=2:DY=1:X2=0:Y2=0
1880 POKE G,X:POKE G+1,Y:POKE G+21,1
1890 B$=" ":X=X+DX:Y=Y+DY:GET A$:IF A$<>
"" THEN 1950
1900 IF X>255 THEN X=255:DX=-DX
1910 IF Y>200 THEN Y=200:DY=-DY
1920 IF X<65 THEN X=65:DX=-DX
1930 IF Y<75 THEN Y=75:DY=-DY
1940 GOTO 1880
1950 IF A$="+" THEN DX=DX+SGN(DX):DY=DY+
SGN(DY):DX=DX-(DX=0):DY=DY-(DY=0)
1960 IF A$="-" THEN DX=DX-SGN(DX):DY=DY-
SGN(DY):DX=DX+(DX=0):DY=DY+(DY=0)
1970 IF A$="+" OR A$="-" THEN 1890
1980 IF A$="■" THEN B=(B+1)AND15:POKE 53
281,B
1990 IF A$="■" THEN E=(E+1)AND15:POKE 53
280,E
2000 IF A$=">" THEN C=(C+1)AND15:POKE G+
39,C
```

it will automatically run the BASIC program in memory (Note: not the Sprite PGM but the BASIC program in memory). In this case it is still your original BASIC program since the sprites were presumably loaded after the end of your program. Thus the first line to be executed will be line 10 again. This time however, since FLAG now has the value 1, the load command will not be executed and the program will carry on.

Replace line 10 with

10 LOAD "SPRITE TABLE",8,1

and see what happens.

One more thing has to be taken care of before your program will run successfully. Since your sprites are loaded after your BASIC program and the start of variables pointer points to the end of your program, as variables are created within your program, they may eventually overwrite your sprites. This can be avoided by changing the start of variables pointer to point to a spot just after your last sprite.

As an example suppose the last sprite in your table is on page 150. It will be stored in memory beginning at location 150 x 64 = 9600 and extending to location 9663. You must change the start of variables pointer to point to any location after 9663. The following line in your program will accomplish this.

20 X=151*64 : HI=INT(X/256) : LO=X-256*HI : POKE 45,LO : POKE 46,HI : CLR

The CLR in line 20 will destroy all variables, so this line should be executed before any variables required by your program are created—preferably immediately after line 10.

One minor annoyance with this setup arises if the program is stopped and run again. Each time this happens, the sprites will be reloaded into memory even though they are already there. This can be avoided by saving the sprite tables along with the BASIC program. To accomplish this, delete lines 10 and 20 from your BASIC program and save the result under some name, say 'ORIGINAL PGM' Then essentially what you do is execute lines 10 and 20 in direct mode and resave

```
2010 IF A$="■" THEN X2=1-X2:POKE G+29,X2
2020 IF A$="■" THEN Y2=1-Y2:POKE G+23,Y2
2030 IF A$=CR$ THEN POKE G+21,0:RETURN
2040 IF A$<>B$ THEN 1890
2050 GET B$:A$=B$:IF B$<>" " THEN 1980
2060 GOTO 1890
2095 REM ******************************
2096 REM *                            *
2097 REM * DISPLAY COLOUR REGISTERS *
2098 REM *                            *
2099 REM ******************************
3000 PRINT LEFT$(V$,20)TAB(34)CO$(E):RET
URN
3010 PRINT LEFT$(V$,21)TAB(34)CO$(B):RET
URN
3020 PRINT LEFT$(V$,22)TAB(34)CO$(C):RET
URN
3030 A1$="RG-0:"+CO$(M0):A2$="RG-1:"+CO$
(M1):IFMR=0THEN A1$="            ":A2$=A1$
3040 PRINT LEFT$(V$,23)TAB(29)A1$:PRINT
TAB(29)A2$"■":RETURN
```

```
1616 OPEN1,8,15,"IO"
1617 OPEN2,8,1,"SCROLL.DATA"
1618 PRINT#2,CHR$(166)CHR$(126);:GOSUB6
000
1619 FOR I=32422 TO 32761
1620 READ J:PRINT#2,CHR$(J);
1621 GOSUB 6000
1622 NEXT
1623 CLOSE2
1624 CLOSE1
1625 END
6000 INPUT#1,A$,B$,C$,D$
6010 IF VAL(A$)=0 THEN RETURN
6020 PRINT"{CLEAR}{DOWN}{DOWN}{REV}DISK
 ERROR "A$
6030 PRINT"{DOWN}"B$
6040 CLOSE2
6050 END
9000 DATA169, 0, 160, 71, 153, 0, 126,
136, 192, 62, 208, 248, 177, 251, 153,
0
9010 DATA126, 136, 16, 248, 96, 160, 71
, 185, 0, 126, 141, 13, 127, 136, 185,
0
9020 DATA126, 141, 12, 127, 136, 185, 0
, 126, 141, 11, 127, 162, 2, 46, 13, 12
7
9030 DATA46, 12, 127, 46, 11, 127, 62,
72, 126, 202, 62, 72, 126, 202, 62, 72
9040 DATA126, 232, 232, 232, 232, 232,
224, 74, 144, 227, 136, 192, 255, 208,
200
9050 DATA162, 71, 189, 72, 126, 157, 0,
 126, 202, 16, 247, 160, 62, 185, 0, 12
6
```

the program. Here are the actual steps.

1. In direct mode, type LOAD "SPRITE TABLE",8,1
—this loads your sprites.

2. Type NEW to fix up the pointers.

3. Type LOAD "ORIGINAL PGM",8
—now your BASIC program and your sprites are in memory. The pointers, however, are not properly set.

4. In direct mode, type
X=151*64 : HI=INT(X/256) :
LO=X-256*HI : POKE 45,LO :
POKE 46,HI :CLR

Now resave this program under a different name and your sprite table will be saved along with this program. All pointers will be properly set too.

Relocating a Basic Program

The above procedure works fine as long as your BASIC program is relatively small (actually BASIC PROGRAM + SPRITE TABLE must be less than 14K). If your program is larger than 14K and requires the use of sprites, another technique is required. This involves storing your sprite tables where BASIC normally begins and moving your BASIC program up in memory.

Here's how it is done.

1. First find a location that is above the last sprite in your table (as in line 20 above). Let's suppose that this value is stored in X.

2. Change the beginning of BASIC pointer to point to X by typing in the following in direct mode.
HI=INT(X/256) : LO=X-256*HI :
POKE 43,LO : POKE 44, HI : CLR

3. The location immediately before the start of the BASIC must be a zero. Therefore type in
POKE X-1,0

4. Now load in your BASIC program as you normally would. It will be stored in memory beginning at location X.

All of the above seems a bit complicated, especially if somebody else is going to be using your program. This can be avoided by having a little BOOT program do all of this for you. Then a user would simply have to load and run the BOOT program. The SPRITE BOOT program is an example of this technique.

ENJOY!!!

```
9060 DATA145, 251, 136, 16, 248, 96, 23
4, 234, 234
10010 DATA 169, 21, 141, 2, 0, 169, 83,
 141, 253, 0, 169, 4, 141, 254, 0
10020 DATA 162, 0, 160, 0, 161, 251, 10
, 72, 169, 46, 144, 2, 169, 81, 145
10030 DATA 253, 200, 104, 192, 24, 240,
 16, 192, 16, 240, 4, 192, 8, 208, 232
10040 DATA 230, 251, 208, 2, 230, 252,
208, 222, 230, 251, 208, 2, 230, 252, 2
4
10050 DATA 173, 253, 0, 105, 40, 141, 2
53, 0, 173, 254, 0, 105, 0, 141, 254, 0
10060 DATA 206, 2, 0, 208, 192, 96
10080 DATA 169, 20, 133, 251, 162, 3, 1
34, 252, 160, 3, 177, 253, 136, 136, 13
6
10085 DATA 145, 253, 230
10090 DATA 253, 208, 2, 230, 254, 198,
252, 208, 237, 198, 251, 208, 231, 169,
 0
10100 DATA 145, 253, 200, 145, 253, 200
, 145, 253, 96, 169, 20, 133, 251, 162,
 3
10110 DATA 134, 252, 160, 0, 177, 253,
200, 200, 200, 145, 253, 165, 253, 208
10115 DATA 2, 198, 254, 198
10120 DATA 253, 198, 252, 208, 235, 198
, 251, 208, 229, 169, 0, 160, 1, 145, 2
53
10130 DATA 200, 145, 253, 200, 145, 253
, 96, 162, 21, 160, 2, 24, 177, 253, 42
10140 DATA 145, 253, 136, 16, 248, 24,
169, 3, 101, 253, 133, 253, 169, 0, 101
10150 DATA 254, 133, 254, 202, 208, 229
, 96, 169, 21, 133, 251, 160, 0, 162, 3
10160 DATA 24, 177, 253, 106, 145, 253,
 200, 202, 208, 247, 24, 152, 101, 253
10170 DATA 133, 253, 169, 0, 101, 254,
133, 254, 206, 251, 0, 208, 225, 96
READY.
```

UMI Software is Making "Home" Work Fun

Wordcraft 20

UMI gives you sophisticated word processing software complete in one package! Wordcraft 20©, with a tutorial tape, contains 8K RAM, a unique automatic mail list feature, and everything else you'll need to create picture-perfect documents. This fully featured system lets you change a character, a word, an entire block of text; and sends encoded electronic mail. With 4-direction scrolling, you see it before you print; and it's compatible with any printer. With Wordcraft 20©, you'll never be at a loss for words again.

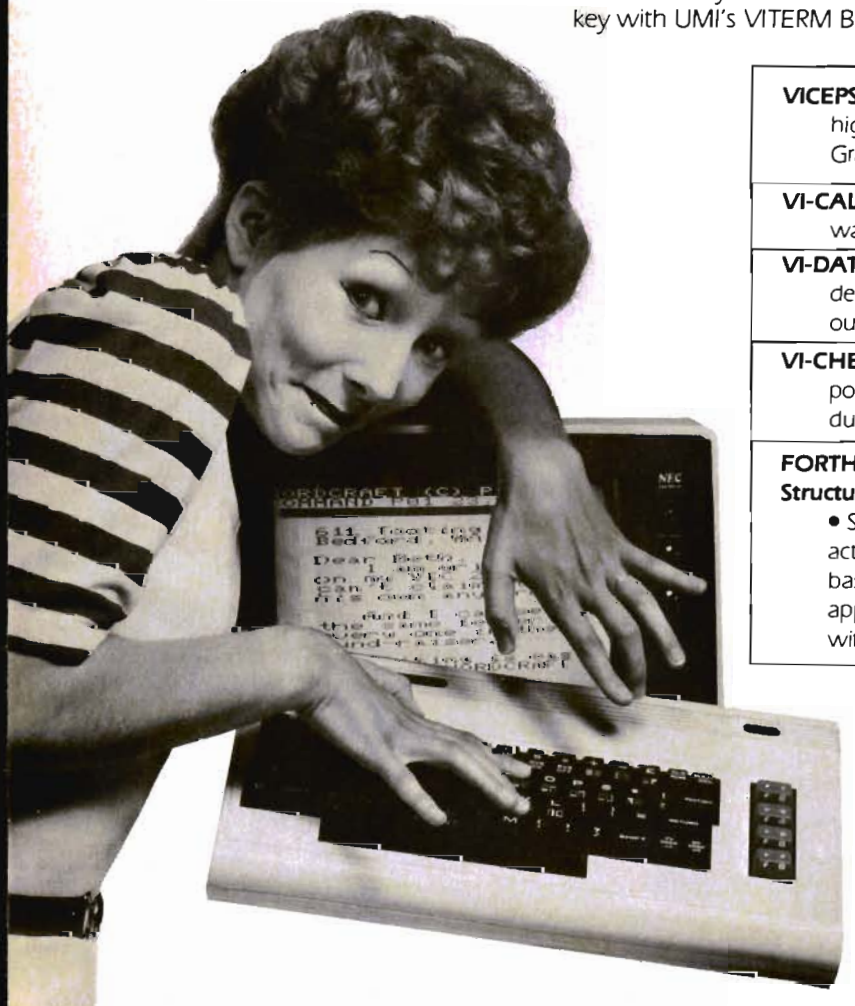
Viterm B

A sophisticated communications program that links you and your VIC™ to the world of information, VITERM B is compatible with virtually any modem. Your access to information banks and services over the telephone system is astonishing. At your fingertips, you'll have UPI news and features, information encyclopedias, discount buying services, the stock market and educational programs. And, VITERM B accesses CompuServe, THE SOURCE, and other similar computer services. You'll be able to send and receive personal electronic mail, set up personal finance programs, make travel reservations — all at electronic speed. The world is yours at the touch of a key with UMI's VITERM B.

BUTI

Improve your BASIC program with UMI's BUTI treatment. Adding 17 new commands to the BASIC language in your computer, BUTI formats the VIC™ to imitate 8K, 3K, or minimum memory configurations. BASIC program errors will stop program execution, list and mark the line of BASIC where the error occurred. Other features are single-step execution, renumbering, block search & replace, block line delete, tape append, and BASIC variable dump.

Simple . . . quick . . . and on command. That's the BUTI treatment for your VIC™.



VICEPS — Connects Epson MX100 or MX80 to your VIC 20™ • Prints high-resolution graphics and character sets using Epson Graftrax • Does formatted BASIC program listings

VI-CALC — 10 memory registers and 4 stacked data • Registers always visible • Math function results visible at a single keystroke

VI-DATA — Powerful data base program on cassette or disk • User-defined screen format • Print screen format • Format print output • Alphanumeric sort

VI-CHECK — Manages checkbook • Lists accounts • Makes deposits • Keeps balance current • Lists transactions • Catches duplicate entries • Features calculator mode

FORTH 20
Structure of PASCAL or COMAL:

• Speed of machine code — 10 times faster than BASIC • Interactive; both a compiler and an interpreter • Transportable — based on FORTH 79-Standard • A language you tailor to your application by adding new commands • Comes complete with an extensive instuction manual and examples.

umi

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

Peek & Poke #8

by George R. Gaukel

In my last article we discussed memory mapping in the VIC. Now we will take a look at how the VIC uses the data at the various locations. First I will define the terms we will be using and some addresses.

VIDEO MEMORY—The mapped area of 1000 bytes contains character pointers or color data. On power-up this area is located at 1024-2023 $400-$7E7.

COLOR MEMORY—The mapped area of 1000 bytes containing COLOR NIBBLES with a byte to byte congruity to the video memory. This area is always located at 56295-57294 $D800-$DB7E.

VIDEO BASE—The mapped area of 2048 or 8000 bytes containing the bit pattern for character or graphic display. On power-up the character ROM is used as the display base. The ROM contains two bases, the graphics character set or the upper/lower case set.

BACKGROUND COLOR REGISTERS—There are four background registers which can contain a color value from 0-15. On power-up BGC0 contains the current background color.

BGC0—53281 $D021
BGC1—53282 $D022
BGC2—53283 $D023
BGC3—53284 $D024

MCM—The multicolor mode control BIT4 at 53270 $DO16. The value is zero on power-up.

ECM—The extended color mode control BIT6 at 53256 $D011. The value is zero on power-up.

BMM—The bit map mode control BIT5 at 53265 $D011. The value is zero on power-up.

There are three character modes and two bit mapped modes.

STANDARD CHARACTER MODE
MCM = ECM = BMM = 0

EXTENDED COLOR CHARACTER MODE
ECM = 1 MCM = BMM = 0

MULTICOLOR CHARACTER MODE
MCM = 1 ECM = BMMM0

STANDARD BIT MAP MODE
BMM = 1 MCM = ECM = 0

MULTICOLOR BIT MAP MODE
BMM = MCM = 1 ECM = 0

Type in, save and then run the CHARS program (it self-destructs). CHARS generates a machine language program at 49152 $C000 which copies the character ROM to 16384 $4000, sets the video page to 31744 $7000 and resets the top of memory pointers. We can now alter the character set to any degree desired and save custom sets to tape or disk using peeks or a monitor.

We will look at the standard character mode first, as this is default on power-up. The video base contains 8 bytes * 256 starting at $6384. The alternate character set starts at 18432 $4800. The first character is '@' which is what we get if we poke a zero to video memory. The second character is 'A' and so on. The eight bytes for each character are mapped in to an 8x8 matrix:

```
16384 ..***..  60 $3C
16385 .**..**. 120 $66
16386 .**.***. 110 $6E
16387 .**.***. 110 $6E
16388 .**.....  96 $60
16389 .**..*..  98 $62
16390 ..****..  60 $3C
16391 ........  00 $00
```

Note that the low bit is to the right and that there is no rotation as the array is displayed. The low bit and bottom row are clear to provide separation. The letter 'A' would be the next eight bytes starting at 16392. We can now redefine keys for special characters such as math characters, rotate characters, or create special graphic characters.

The extended color character mode allows us to select up to four different background colors for each individual character, with each character being available in any of the 16 colors. The trade-off is that only the first 64 8x8 characters in the video base can be used. This is because the high two bits in the video memory are used as color pointers. The character color is still determined by color memory as in the standard mode.

BIT7	BIT6	
0	0	BGC0 0 + CHARACTER VALUE 0-63
0	1	BGC1 64 + CHARACTER VALUE 0-63
1	0	BGC1 128 + CHARACTER VALUE 0-63
1	1	BGC2 192 + CHARACTER VALUE 0-63

To set the ECM bit enter X = PEEK(53265)OR64: POKE53265,X. We can now use the ECM mode by poking color values to the background registers. If we poke a zero to the video memory, we should get '@' BGCO. Poke 193 would get an 'A' with BGC3. This is a nice mode for making widows for text applications. To clear the ECM bit enter X = PEEK(53265)AND191: POKE53265,X.

The last character mode is the multicolor character mode. To set the MCM bit enter X = PEEK(53270) OR16:POKE53280,X. In this mode we can mix 8x8 characters with 4x8 graphics. If the high bit of the color memory is clear (BIT3) the eight bytes are displayed as a 8x8 character. As the high bit is used as a toggle, only the low eight colors are available for individual characters. If the high bit in

the color nibble is set the eight bytes are displayed in a 4x8 graphics mode. We still use the full eight bytes of the base but the bits are now used as a paired color pointers to color registers. Each bit pair will display the indicated color as two dots.

BIT	PAIR	COLOR
0	0	BGCO
0	1	BGC1
1	0	BGC2
1	1	COLOR NIBBLE

This mode allows user graphic character of up to four colors. The background colors will be common to all characters and one of eight colors available for a any individual character using the color nibble. To clear the ECM bit enter X=PEEK(53270)AND 239:POKE53270,X.

Think of the above character modes as 256 tiny sprites that can be put anywhere on the screen any number of times. Their location in the video base has no bearing on where they are displayed on the screen. The bit mapped modes however have a one to one bit congruity with each dot on the screen and require 8000 bytes of video base (320*200/8). In the standard bit mapped mode the color is determined by the video memory pointer. The color nibbles are not used. The first video pointer controls the color of the first 8x8 mapped array. If a bit is set the color will be that of the hi nibble of the video pointer. A clear bit will use the low nibble of the video pointer for its color value. Any of the 16 colors can be used in either nibble. So, within an 8 byte block we can have two colors. There will be 1000 8 byte blocks with their color controlled by the 1000 video memory bytes.

The last mode is the multi-colored bit mapped mode. The horizontal resolution is half that of the standard mode because we again go to paired color pointers. The 8 byte block now maps to 4x8 with each byte containing 4 color pointers.

BIT	PAIR	
0	0	BGC0
0	1	HIGH NIBBLE OF VIDEO POINTER
1	0	LOW NIBBLE OF VIDEO POINTER
1	1	COLOR NIBBLE VALUE

With the reduced resolution we can now have a standard background color or any of three other colors. The background color will be common to the total display. The other colors can be changed as we move from one eight byte block to another. This mode will allow adequate resolution for most applications.

The mapping of the bit mode blocks is the same as for the character blocks. If we wanted to put text on a mapped screen we could copy the character as it appears in the character ROM, and it would appear normal in the standard bit mode. We would have to copy eight bytes for each character used or we could set up software character generators.

I expect to see cartridges or software very soon, that will make the graphic capabilities available to BASIC programs using standard commands such as DRAW, PAINT, PLOT and such. Until then, we can still get some very useful results for applications and games.

In the CHARS program I copied the character ROM into the middle of BASIC RAM so we could get a good look at it. We could now copy it to the RAM under the KERNAL ROM at $E000 and put the video page in the $C000 area. This would leave the BASIC work space free. The VIC always looks at the RAM in this area and not at the ROM. This area could also be used for bit mapped displays.

CHARS

```
100 FOR A=49152TO49234:READ D:POKE A,D:NEXT
110 DATA 216, 169, 208, 133, 252, 169
120 DATA  64, 133, 254, 169,   0, 168
130 DATA 133, 251, 133, 253, 120, 169
140 DATA  51, 133,   1, 177, 251,  72
150 DATA 169,  55, 133,   1,  88, 104
160 DATA 145, 253, 141,  39,   4, 200
170 DATA 208, 234, 230, 252, 230, 254
180 DATA 165, 252, 201, 224, 208, 224
190 DATA 169, 255, 141, 131,   2, 133
200 DATA  55, 169,  63, 141, 132,   2
210 DATA 133,  56, 173,   0, 221,  41
220 DATA 252,   9,   2, 141,   0, 221
230 DATA 169, 240, 141,  24, 208, 169
240 DATA 124, 141, 136,   2,  96
250 SYS49152 : NEW
```

Bytes, Bits and Binary

by Tim Parker
Canada

Most computer users are acquainted with the "high level" languages such as BASIC, FORTRAN, or PASCAL, and consider these to be powerful. All have one common feature that most people do not realize: they are all converted by the computer to the same fundamental units: binary digits.

Binary programming is a feat that requires an extraordinary amount of patience, and a not inconsiderable amount of fortitude. However, the binary language is the one that is ultimately used by every computer, every computer language, and every operating system in the world. It is perhaps a symptom of our computer society that a large percentage of the population can consider themselves to be "computer literate," and yet be totally lost when confronted with the strings of zeros and ones of binary.

This article is not intended to make binary programmers of you: such a creature would be very unusual. Its purpose is to give a brief insight into binary, and the way a computer can use the binary strings for the manipulation of data.

On the road to the "low level" languages such as binary, machine language programming comes into play. Machine language is in fact a glorified binary programming. By assigning a mnemonic code to certain functions that the machine can understand directly, a fast, reliable program can be constructed.

Two other forms of simple codes are used by computers: octal and hexadecimal. As the name suggests, octal is based on eights, and hexadecimal on sixteens. The root of this is not difficult to determine.

Most computers in use in the home are eight bit machines: they use "blocks" of eight binary digits to control the functions of the computer. Thus, having a counting system based on eights seems logical. (Our ten-based system arose because of the multitude of ten-based objects around us—our fingers are the prime example.)

Counting in octal is not difficult, but will not be delved into here, as it is rarely found on home machines. Far more common is hexadecimal. Hexadecimal numbers are based on a combination of two eight binary blocks. (An eight binary block is called a byte, while each binary digit is a bit: There are eight bits to a byte.)

The usual counting system is used up to hexadecimal 9 (hex numbers are symbolized by the letter H after them, such as 5H). After nine, the problem arises in finding a one digit symbol for what we normally think of as a two digit number. The answer is to use the first six letters of the alphabet. Thus, in binary, the following letters are used for hexadecimal ten though fifteen:

10 = A
11 = B
12 = C
13 = D
14 = E
15 = F

When counting in hexadecimal, each digit in a number represents a power of 16. Just as 157 in ten-based systems means 1 hundred unit (ten to the second power) plus 5 ten units (ten to the first power) and 7 one units (ten to the zero power), in hexadecimal, 3C2 represents 3 sixteen to the second power units plus 12 sixteen to the first power units plus 2 sixteen to the zero power units.

All this takes a little getting used to. Seeing the number 5H obviously represents 5 in the decimal system. FH, from the rules above, represents 15 decimal, but the sight of 1B15H can be a little awe inspiring at first sight. But working it out as done above shows it to be 457 decimal. [1 x 256 + 11 x 16 + 5 x 1].

What use is hexadecimal? When programming machine language, many of the lines are displayed as hexadecimal codes representing different functions. Also, memory locations are indexed by hexadecimal. So while a BASIC programmer may not care what the hex code for a screen memory location is, the chances are the machine language programmer will have very little choice but to know it.

Binary programming, on the other hand, doesn't have to be known by anyone: it serves no major programing function. It is important in that it is how the machine does everything.

Each bit in the computer can be either on or off (empty or full). It is convenient to think of an off bit as possessing the value O, while an on bit has the value one.

A byte (eight bits together) can then have any combination of the bits on or off, for a total of 255 different arrays. To count in binary, the bits are turned on or off from the right to left:

DECIMAL	HEXADECIMAL	BINARY
0	0	0
1	1	1
2	2	10
3	3	11
4	4	100
5	5	101
6	6	110
7	7	111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100

13	D	1101
14	E	1110
15	F	1111
16	10	10000
17	11	10001
18	12	10010

and so on. The space filling method is easily determined by examination of the series above. Extension to any number up to 255 is simply a matter of filling in the ones where needed. (255 is FFH.)

Addition of two binary numbers can then be seen as simply a matter of positioning the bits so they correspond with each other, and adding together. Where a two would appear, a zero is placed, and one carried. As long as the total doesn't exceed 255, there is no problem with this method.

The problem arises when subtraction is considered. Subtraction is to take the negative of one number and add it to the other. This could be done by using one of the bits as a symbol of whether the number is positive or negative, but that leaves only seven bits left for number crunching. That limits the maximum number to 127! Obviously a better scheme must be found.

The better scheme is called two's complement. The reason will be obvious in a moment. If all bits were reversed in a byte (i.e. 00010010 would become 11101101) then subtraction would work the same way as addition seen above. A one is added to the result to convert the zeroes to ones, and vice versa. This reversal of bits is called one's complement. A problem arises with one's complement when one number and its complement are added. For example, 16 + (116) should total zero. If the bits are added together, though, an extra one appears at the left of the byte:

00010000	16
+11101111	−16
11111111	0

For one's complement, now, a one is added to the result:

```
 11111111
+00000001
----------
1000000000
```

If the extra digit is ignored, the result (00000000) is exactly what we want, but the extra digit cannot be just discarded and ignored. The extra digit has no place in memory, and so the one's complement method will not suffice. Two's complement grew out of this failure. Simple stated, two's complement is formed by adding one to the one's complement!

Consider a calculation similar to the one above, where the two's complement of twenty-five will be found:

```
 11100110 one's complement of 25
+00000001
----------
 11100111 two's complement of 25
```

A few examples of arithmetic here will help illustrate the use of two's complement. Addition is the same as ever:

00000011	3
+00000101	+5
00001000	8

while subtraction requires use of two's complement:

00000011	3
+11111011	−5
two's complement of five	
11111110	−2

The result is obtained by taking the two's complement of the number (i.e. adding one to the one's complement):

00000001	one's complement of 11111110
+00000001	adding 1
00000010	= 2 in decimal

As the true result was 11111110, this represents a negative number, so the answer is −2!

This obviously is a very elaborate way to calculate answers to questions! However, it does work, and is the only way a machine with only two states (on/off) can do it. A few more elaborate proposals have been made, but two's complement remains in use. Although it takes a lot of getting used to, it does on study appear to be an elegant method of taking the problem.

Luckily, programmers don't need to know binary. But it's nice to have an inkling what goes on deep in those chips.

WHY DIDN'T I THINK OF THAT?

by Howard Rotenberg
Toronto, Canada

Have you ever been using a routine for a long time assuming that it was common knowledge? I have met a lot of people using useful routines that I wished that I had in my little bag of tricks. The funny thing about them is that they are usually so simple the question arises, Why didn't I think of that? These were the exact words that a friend of mine used the other day that inspired me to pass on these few routines. NOTE: Usually he is the one to figure them out first.

The routine that he spoke about is the one I will present first. This small routine may be used in a program to see if the printer is turned on before trying to print text. (See program 1)

I have used the word "home" to represent the home key on the keyboard. As you see, the routine is simple and may be easily put into any program. The crucial lines are 15 and 20. After the printer is opened, we do a simple print#4. Line 20 checks the status at this point and it should be zero if the printer is on. If it is not, it looks back and tries again. The program loops until the printer is finally turned on in which case you are graciously thanked. You may, of course put in any type of error detection you wish (ie: flashing messages), however, I chose to keep it simple for the example. One strange occurance was that we tried the routine with the IEEE cable disconnected expecting to see a DEVICE NOT PRESENT error, however, we did not. This one I will let you figure out.

A short but useful routine to read sequential files follows. I have used a small machine language routine to read the file and will explain exactly what it does and its BASIC equivalent.

```
10 FOR J = 0 TO 22
20 READ A
30 POKE 634 + J,A
40 NEXT J
50 INPUT "FILENAME";F$
60 DOPEN#1,(F$)
70 SYS 634
80 DCLOSE
90 GOTO 50
95 DATA 162, 1, 32, 198, 255, 32,
   228, 255, 32
97 DATA 210, 255, 166, 150, 208,
   2, 240, 244
99 DATA 162, 1, 32, 166, 242, 96
```

Line 70 jumps to the machine language routine loaded by the first four lines. The BASIC equivalent to this line is:

```
70 GET #1,A$: PRINT A$;: IF
   ST XX 64 GOTO 70
```

The main reason for using the machine language is of course the speed at which it will read and display the information. You may easily substitute the BASIC line 70 to see the different yourself. Since the routine is just loaded as data statements that don't mean too much to a lot of users, program 2 will show exactly what the routine is.

The two JSR instructions ($FFC6 and $F2A6) are ROM dependent for BASIC 4. The other routines are Kernal routines that may be used on all Commodore computers. To find the proper routines for the VIC 20 or the Commodore 64, you may just look in any memory map. I have the maps but without the computers to test it on I would rather not print the addresses in case of error.

Another routine I have found many opportunities to use is one that takes TI$ and converts it into a regular everyday 12 hour clock. It even displays AM or PM. It's not that I am opposed to the 24 hour military clock but unless you are in the service or a hospital (hospitals like to use the 24 hour clock), the more familiar 12 hour clock seems more friendly. (See program 3)

I have included a return statement in line 70 since this routine lends itself to be called as a subroutine. I have found that displaying the time in the more common way adds less confusion to an end user of one of my programs.

I imagine I could go on for a long time sharing these small routines as I am sure we all could. Rather than doing that I would like to end off with just a few very small but extremely useful ones that I find very handy.

The version of MICROSOFT BASIC that comes on the Commodore computers lacks an extremely useful logical operation. They have included the AND, OR and NOT function (even though the NOT function is NOT what you would expect it to be), however, unless you get into machine code you do not have an exclusive or function. The truth table for this function is as follows:

```
0:0 = 0
0:1 = 1
1:0 = 1
1:1 = 0
```

To EOR or as some languages present it (XOR) the variable A with B the following may be done: ((A) AND (NOT (B))) OR ((B) AND (NOT(A)))

Next we will round off the number (N) to (D) decimal places:

```
10 INPUT D
20 INPUT N
30 N = INT(N * 10 ! D + .5)/10 ! D
40 PRINT N
```

I have used an exclamation mark instead of the exponent or up arrow sign in line ten, since my method of writing this text does not include that character. If you use the routine, please

Program 1

```
5 REM *** TEST AND WAIT FOR PRINTER ***
10 OPEN 4,4
15 PRINT#4
20 IF ST XX 0 THEN PRINT "/HOME/TURN ON PRINTER": GO TO 15
30 PRINT "THANKYOU"
40 REM *** THE REST OF THE PROGRAM ***
```

Program 2

```
        * = $027A    ;START OF ML (634 DECIMAL)
        LDX #$01     ;GET THE FILE NUMBER
        JSR $FFC6    ;SET THE INPUT DEVICE
GETCH   JSR $FFE4    ;GET A CHARACTER
        JSR $FFD2    ;WRITE ONE CHARACTER
        LDX $96      ;GET THE STATUS VALUE (ST)
        BNE CLOSE    ;IF NOT ZERO THEN GOTO END OF ROUTINE
        BEQ GETCH    ;GO BACK FOR ANOTHER CHARACTER
CLOSE   LDX#$01      ;GET THE FILE NUMBER AGAIN
        JSR$F2A6     ;CLEAR THE CHANNEL
        RTS          ;BACK TO BASIC
        .END
```

Program 3

```
10 INPUT "TIME";TI$
20 OI$ = ""
30 IF VAL (LEFT$(TI$,2))) = 12 AND VAL (MID$(TI$,5,2))) = 1 GOTO 80
40 LK$ = "AM.": IF VAL (LEFT$(TI$,2)) (= 9 THEN OI $ =
   MID$ (TI$,2,1): GOTO 60
50 OI$ = MID$ (TI$,1,2)
60 OI$ = OI$ + MID$ (TI$,3,2) + LK$
70 PRINT "THE TIME IS" OI$: RETURN
80 IF VAL (LEFT$(TI$,2)) = 12 THEN OI$ = LEFT$ (TI$,2): LK$ =
   "PM.": GOTO 60
90 OI$ = STR$ (VAL(LEFT$(TI$,2)) – 12): LK$ = "PM.": GOTO 60
```

Program 4

```
10 L = 0: FOR J = 1TO4: L% = ASC (L$): L% = L% – 48 + (L%)64)*7
20 L$ = MID(L$,2): L = 16*L + L%: NEXT: PRINT L
```

Program 5

```
10 L = L/4096: FOR J = 1TO4: L% = L: L$ = CHR$(48 + L% – (L%)9)*7
20 PRINT L$;:L = 16*(L – L%): NEXT
```

Program 6

```
10 FOR J = 634 TO 656
20 READ H$: M$ = LEFT$(H$,1): L$ = RIGHT$(H$,1)
30 POKE J, (ASC(M$) – 48 + ((M$)"9")*7))*16 + ASC(L$) – 48 + ((L$)"9")*7))
40 NEXT J
50 DATA A2, 01, 20, C6, FF, 20, E4, FF, 20, D2, FF
60 DATA A6, 96, D0, 02, F0, F4, A2, 01, 20, A6, F2, 60
```

use the proper key as I can guarantee you that the exclamation mark will not work.

In an earlier example I used a basic loader to poke a small machine subroutine into memory. These numbers, however, had to be converted to decimal first so here are two one liners to do conversions for us. First we submit the routine where our hex number is in the variable L$, and is returned in L as a decimal number. (See program 4)

Now we submit the variable L to this decimal to hex routine. (See program 5)

The two conversion routines may be put into one line, however, I decided for the sake of clarity to display them as two.

Lastly, if you do not want to convert a hex number to decimal but prefer to list it in data statements as hex and still poke it into memory, then the final routine is for you. (See program 6)

The data statements contained the same sequential file reading routine that I used earlier.

Conclusion:

We over the years seem to acquire a number of subroutines that we like to use in our programs. Some of course are much more difficult than others, however, even if we don't fully understand them, with the proper instructions, we may all benefit from them. There is a limit on the number of ways that one can implement the same function using any one given premise. I am sure that there are a lot of routines out there being used every day that do the same as the few I have shown. I hope mine will be of use to some of the readers and if so, please let us in on your tricks that you assume that everyone else already knows. If you do, I am sure that just like that friend and myself, a lot more people out there will benefit greatly while saying: "Why didn't I think of that?"

Debugging & Fixes of Programs

Commander has dedicated this column exclusively to the Debugging and Fixes of Programs. We want to keep our readers informed of any changes or mistakes in programs we publish. This column is designed for that purpose only. If you as a reader find any bugs, please let us know. —Editor

FIXES

Volume 1, Issue 2

Page 16, first paragraph: 50 octal converts to 40 decimal.

Page 20, bottom, line 0020: WHILE I X 5 10

Page 21, top, line 0050: UNTIL I X 10

Page 21, third paragraph: A8 (6.2) would be IS

User Clubs

New Hampshire
TBH VIC-NIC CLUB
P.O. Box 981
Salem, NH 03079
Contact—J. Newman
Publication—VIC-NIC NEWS
Interests—VIC-20 Exclusively

Kentucky
The Commodore Connection
1010 South Elm
Henderson, Kentucky 42420
Contact—Jim Kemp (502) 827-8153
Interests—VIC, CMB 64, PET

Ohio
SW OHIO VIC USERS CLUB
659 Carthage Ave.
Cincinnati, Ohio 45215
Contact—Tom E. Harris 761-7510

Indiana
The VIC Indy Club
P.O. Box 11543
Indianapolis, IN 46201
Contact—Linda Kropzer
(317) 878-3342

So. Dakota
VIC/64 USERS CLUB
203 East Sioux Ave.
Pierre, So. Dakota 57501
Contact—Larry J. Lundeen
(605) 224-4863

Illinois
The Fox Valley PET User's Club
833 Willow St.
Lake in the Hills, IL 60102
Contact—Art Dekneef (312) 658-7321

Washington
A.T.S. VIC-20 Computer Club
7906 34th Ave. S.W.
Seattle, WA 98126
Contact—Ken Gazaway (206) 935-2697
Publication—For VIC-20 only

Central Washington Commodore User's CLub
1222 So. 1st St.
Yakima, WA 98902
Contact—Bob Wood or Tim McElroy

Commander Dealers

Given here, in zip code order, is a partial list of the Charter Dealers who will be carrying the COMMANDER. We will provide updates for this list in following issues as a service to provide our readers with a local source at which they will find information, hardware, or software for their Commodore Computers.

U.S.A.

Puerto Rico

The Micro Computer Store
1408 Ave. Jesus T. Pinero
Rio Piedras, P.R. 00921
(809) 781-0350
Manager-Owner: Julio C. Martinez

Computech Ltd.
214 Derby Street
Salem, MA 01970
(617) 741-1724
Manager-Owner: Tim Bush

Omicron Corporation
1416 Providence Highway
Norwood, MA 02062
(617) 769-6867
Manager-Owner: Steve Gavrilles

Rhode Island

International Computer Services
165 Oyerville Ave.
Johnston, RI 02919
(401) 273-1001
Manager-Owner: Steve Lablanc

New Hampshire

Compu-Craft, Inc.
17 Dunbar St.
Keene, NH 03431
(603) 357-3901
Manager-Owner: Richard Bishop

Echo Consulting Services
P.O. Box 1199
Conway, NH 03818
(603) 447-5455
Manager-Owner: George Epotien

Maine

Maine Micro Systems, Inc.
55 Center St.
Auburn, ME 04210
(207) 786-0696
Manager: Nancy Lecompte

Vermont

Computeam
205 Dorset St.
S. Burlington, VT 05401
(802) 862-2802
Manager-Owner: Mark Robinson

Connecticut

Multi-Business Computers Inc.
28 Malborough Street
Portland, CT 06480
(203) 342-2747
Manager-Owner: Bob Stasko

New Jersey

Micro Computer Services
61 Mountain Blvd.
Warren, NJ 07060
(201) 561-3111
Manager-Owner: Jerry Prevete

Computer Workshop
1200 Haddenfield Rd.
Cherry Hill, NJ 07013
(609) 665-4404
Manager-Owner: Charles Kolbe

Computerability, Inc.
441 Route 23
Pomton Plains, NJ 07444
(201) 835-0688
Manager-Owner: Dennis Mull

Wayne Computer Store
1459 Route 23
Wayne, NJ 07470
(201) 628-7318
Manager-Owner: Rick Delti

Software City
147 N. Kinderkamack Rd.
Montvale, NJ 07645
(201) 391-0931
Manager-Owner: C.M. Hatfield

Computer Workshop
1200 Haddenfield Rd.
Cherr Hill, NJ 08034
(609) 665-4404
Manager-Owner: Chris Kolbe

Electronic Service Assoc., Inc.
Route 541 N.
Medferd, NJ 08055
(609) 953-1200
Manager-Owner: John Handfield

Software-n-Such
Warren Plaza, Route 130
E. Windsor, NJ 08520
(609) 443-8984
Manager-Owner: Paul Hammer

BB/The Computer Store
216 Scotch Rd.
Trenton, NJ 08628
(609)883-2050
Manager-Owner: Barry Brown

New York

CompTek
90 John Street
New York, NY 10038
(212) 962-6131
Manager-Owner: Frances Banks

Computer Emporium
37 North Street
Middletown, NY 10940
Manager-Owner: Kate Honders

B.C. Communications, Inc.
World Wide Electronics Dist.
207 Depot Rd.
Huntington St., NY 11746
(516) 549-8833

Computer Headquarters
1245 Middle Country Road
Selden, NY 11784
(516) 698-9373
Manager-Owner: James & Kathy Lyon

Dealers continued . . .

Upstate Computer Shop
1823 Western Ave.
Albany, NY 12203
(518) 456-3019
Manager-Owner: William J. Smith

Future Distribution
Trimex Bldg.—Route 11
Mooers, NY 12958
(514) 861-2831
Manager-Owner: Phillippe Faure

Durmac Cash Registers
1628 Erie Blvd. E.
Syracuse, NY 13210
(315) 472-4531
Manager-Owner: William McCarthy

Personal Computers, Inc.
3251 Bailey Ave.
Buffalo, NY 14215
(716) 832-8800
Manager-Owner: Frank C. Smeirciak

Pennsylvania

One Stop Computer Shope
65 N. 5th St.
Lemoyne, PA 17043
(717) 761-6754
Manager-Owner: Joanne Wright

Micro Age Computer Store
1352 Tilghman St.
Allentown, PA 18102
(215) 434-4301
Manager-Owner: Ed Eichenwald

Maryland

Professional Micro Service
100 W. 22nd St.
Balto, MD 21218
(301) 366-0010
Manager-Owner: James A. Breen

Tri-State Computers
1504 S. Salisbury Blvd.
Salisbury, MD 21801
(301) 742-2020
Manager-Owner: Tom Weiland

Virginia

Virginai Micro Systems
13646 Jeff Davis Highway
Woodbridge, VA 22191
(703) 491-6502
Manager-Owner: Suart Mitchell

CALPRO—The World of Computers
3119 Waterlick Road
Lynchburg, VA 24502
(804) 237-3825
Manager-Owner: Walter Leroy Ashley

West Virginia

Computer Associates, Inc.
113 Hale St.
Charleston, WV 25301
(304) 344-8801
Manager-Owner: Jeff Knapp

North Carolina

The Program Center
3400A W. Wendover Ave.
Greensboro, NC 27407
(919) 855-8667
Manager-Owner: Rupert Fenequito

Piedmont Microsystems Inc.
Route 3 Box 150 H
Frazier Professional Bldg.
Newton, NC 28658
(704) 465-3600
Manager-Owner: Lorne Machael

Bob West Computers
54 West Main St.
Brevard, NC 28712
(704)883-2595
Manager-Owner: Sylvia West

Georgia

Cardinal Computers
903 N. Gleenwood
Dalton, GA 30720
(404) 226-0502

Florida

COMPUTECH
1415 Timberlane Rd.
Tallahassee, FL 32312
(904) 893-1743
Manager-Owner: Dan Evans

Random Access Computers
296 Eglin Parkway
Ft. Walton Beach, FL 32548
(904) 862-7763
Manager-Owner: Joanne Dodd

Florida Book Store
1614 West University Ave.
Gainesville, FL 32604
(904) 376-6066

Skippers Inc.
217 S.E. 1st Ave.
Ocalm, FL 32671
(904) 732-3221
Manager-Owner: David Lee Skipper

Osceola Computer
1300 Dakota Ave.
St. Cloud, FL 32769
(305) 892-1501
Manager-Owner: Raymond Barrieau

Sigma Systems of Orlando
590 N. Semoran Blvd.
Orlando, FL 32807
(305) 273-2434
Manager-Owner: Tom Clance

Computer Specialties, Inc.
701 E. Lincoln Ave., P.O. Box 1718
Melbourne, FL 32901
(305) 725-6574
Manager-Owner: Otis P. Lutz

Micro Byte, Inc
13710 SW 56 Street
Miami, FL 33175
(305) 385-2108
Manager-Owner: Ed Silverman,
Lyman Conover

Focus Scientific
224 N. Federal Highway
Fort Lauderdale, FL 33301
(305) 462-1010
Manager-Owner: M. Rienhardt

The Software Connection
5460 N. State Rd. 7, Suite 108
Ft. Lauderdale, FL 33319

Business Machines
2821 Pinewood Ave.
West Palm Beach, FL 33407
(305) 655-4730
Manager-Owner: Robert Frazier, Jr.

The Software and Computer Store
1506 Gulf-to-Bay
Clearwater, FL 33515
(813) 442-8803
Manager-Owner: Charles Kautz

Alabama

Tricelin Corporation
Route 1, Box 128
Bankston, AL 35542
(205) 689-4999

Tennesee

American Computer Co.
1004 8th Ave. S.
Nashville, TN 37203
Manager-Owner: Jane Maggard

Metro Computer Ctr.
416 W. Main St.

Chattanooga, TN 37402
(615) 875-6676
Manager-Owner: Wayne F. Wilson

Mississippi
Sunrise Persons Supplies
901 So. John St.
Corinth, MS 38834
(601) 287-4721
Manager-Owner: Felex Gathings

Kentucky
All Business Computers
Suite C-2317 Versailles Rd.
Lexington, KY 40504
(606) 253-2545
Manager-Owner: Bud Wilson

Stowehuewge Computer, Inc.
2026-29th St.
Ashland, KY 41101
(606) 359-0545

Ohio
Earthrise Micro Systems, Inc.
562 W. Central Ave.
Delawar, Ohio
(614) 363-1100
Manager-Owner: John Kessler

Office Mart, Inc.
1151 East Main St.
Lancaster, OH 43130
(614) 687-1707
Manager-Owner: Pat Blake

The Computer Store of Toledo, Inc.
18 Hillwyck Dr.
Toledo, OH 43615
(419) 535-1541
Manager-Owner: Al and Jackie Miller

Computer Corner Inc.
5104 Mayfield Rd.
Lyndhurst, OH 44124
(216) 423-5010
Manager-Owner: Ross Black

Computer Showcase
5855 Youngston-Warren Rd. SE
Niles, OH 44446
(216) 652-2571

Waltz Photo
438 Sixth St.
Canton, OH 44701
(216) 455-9421
Manager-Owner: Brad Zupp

Wards Computers, Inc.
868 Ohio Pike
Cincinnati, OH 45245
(513) 752-2882
Manager-Owner: Carl Ward

Indiana
Allan's Jewelry & Loan Co.
130 E. 10th St.
Anderson, IN 46016
(317) 642-7978
Manager: Jerry Rubenstein

McCarels Computers
1204 Meridian Plaza
Anderson, IN 46016
(317) 643-2662

AVC Corporation
2702 Applegate
Indianapolis, IN 46203
Manager-Owner: Brent Enderle

Impair
342 Bosart
Indianapolis, IN 46201
(317) 353-9947
Manager-Owner: Fred Imhausen

A Computer Store
2140 N. Mithoefor Rd.
Indianpolis, IN 46229
(317) 898-0331
Manager-Owner: Skip Robbins

Computer People
900 Highway 212
Michigan City, IN 46360
(219) 879-8557
Manager-Owner: Harry Hopkins

Computer Corner
6722 E. State Blvd.
Fort Wayne, IN 46815
(219) 749-8338
Manager-Owner: Tom Kutina

Custom Software
3197 South 3rd Place
Terre Haute, IN 47802
(812) 234-3242
Manager-Owner: Vicki McEntaffer

Michigan
Micro Station Inc.
24484 W. 10 Mile Rd.
Southfield, MI 48034
(313) 358-5820
Manager-Owner: Jerry Goldberg

Comm Data
320 Summit
Milford, MI 48042
(313) 685-0113

Roseville Computer
25929 Gratiot
Roseville, MI 48066
(313) 772-0760
Manager-Owner: Tom Potter

Allen Park Computer Center
7000 Roosevelt, Suite 109
Allen Park, MI 48101
(313) 383-8254
Manager-Owner: Sam Noble

Haney's Stereo, Inc.
15270 Gratiot
Detroit, MI 48205
(515) 752-8845
Manager-Owner: Paul M. Paul

Computer Mart
915 S. Dort Hwy.
Flint, MI 48503
(313) 234-0161
Manager-Owner: Pat McCollem

Computer Connection
444 W. Maumer Street
Adrian, MI 49221
(517) 265-7872
Manager-Owner: Ron Gamling

Newman A/V Communications, Inc.
400-32nd Street SE
Grand Rapids, MI 49508
(616) 243-3300

Computers and More
2915 Dretom
Grand Rapids, MI 49508

Computer Tutor
502 E. Front
Traverse City, MI 49684
(616) 941-5320
Manager-Owner: Caroline Garrick

Iowa
Micro Computer Applications
111 E. Church St.
Marshalltown, IA 50158
(515) 752-8845
Manager-Owner: Harold Montover

Gronert Computers, Ltd
4505 Forest Ave.
Des Moines, IA 50311
(515) 255-0618
Manager-Owner: Frank Gronert

Dealers continued . . .

Cosmos Computers
1721 Grant Street
Bettendort, Iowa 52722
(319) 355-2641
Manager-Owner: Paul Rung

Wisconsin

Majic Business Systems
3519 W. Wanda Ave.
Milw, WI 53221
(414) 282-8072
Manager-Owner: Dennis Woitekaitis

Computerland of Madison
6625 Odana Rd.
Madison, WI 53719
(608) 833-8900
Manager-Owner: James Sullivan

South Dakota

Computerland Rapid City
738 St. John St.
Rapid City, SD 57701
(605) 348-5384
Manager-Owner: John Mattson

Illinois

The Software Store, Inc.
1767 Glenview Rd.
Glenview, IL 60025
(312)724-7730
Manager-Owner: David Pokritis
and Jeff Rayer

Digital World
711 Army Trail Rd.
Addison, IL 60101
(312) 628-9222
Manager-Owner: Sam Gunda

B-A Computer Sys.
2 N. Batavia Ave.
Batavia, IL 60510
(312) 879-2350
Manager-Owner: Robert Appel

Softwareland, Inc.
420 West 75th St.
Downers Grove, IL 60516
(312) 852-6340
Manger-Owner: Maureen Quinn

Rozel Industries, Inc.
7360 N. Lincoln Ave.
Lincolnwood, IL 60646
(312) 675-8960
Manager-Owner: Fred Whitlock and
Becky Kowalsky

Fisher Scientific
4901 W. Lemoyne Ave.
Chicago, IL 60651
(312) 378-7770
Manager-Owner: A.C. Heidrich

Kappel's Computer Store
125 E. Main
Belleville, IL 62220
(618) 277-2354
Manager-Owner: Tom Kappel

Data Plus, Inc.
1706 Broadway
Quincy, IL 62301
(217) 222-65602
Manager-Owner: James Moore

Missouri

Micro Age Computer
11413 Olive Blvd.
St. Louis, MO 63141
(314) 567-7644
Manager-Owner: Norm Fisher

Common Wealth Computers
5214 Blue Ridge Blvd.
Kansas City, MO 64133
(816) 356-6502
Manager-Owner: Dick York

Kansas

Computer Business Machines
Officenter 357 S. Lulu
Wichita, KS 67211
(316) 267-1150
Manager-Owner: Mrs. R. Santoscoy

Nebraska

Hobby Town
220 N. 66th Street
Lincoln, NB 68505

Central Office Equipment
2020 Central Ave.
Kearey, NB 68847
(308) 234-2515
Manager-Owner: Byron Hanse

Louisiana

The Computer Center
111 C Rena Drive
Lafayette, LA 70503
(318) 988-2478
Manager-Owner: Robert Jones

Texas

Taylor Computer Systems
949 Melbourne Road
Hurst, Texas 76503
(817) 284-5251
Manager-Owner: Mike Taylor

Computer Home
431 East Ave. C.
San Angelo, TX 76903
(915) 653-7488
Manager-Owner: Brent DeMoville

Texas Technical Services
3115 W. Loop S., #26
Houston, TX 77027
(713) 965-9977
Manager-Owner: Phil Ray

The Computer Experience
125 Southbridge
San Antonio, Texas 78217
(512) 340-2901
Manager-Owner: Carolyn Roberts

Software-N-Things
2141 W. Anderson Lane
Austin, TX 78757
(512) 451-4347
Manager-Owner: John Krieg

Professional Computer Associates
5326 Cameron
Austin, TX 78723
(512) 459-1220
Manager-Owner: Steve Derosa

Computerland of Amarillo
2300 Bell St.
Amarillo, TX 79106
(806) 353-7482
Manager-Owner: Mark Trowbridge

Colorado

Whole Life Distributors
965 Washington St. #6
Denver, Co. 80203
(303) 861-2825
Manager-Owner: Tom Tarbart

Zero Page, Inc.
2380 Naegele Rd.
Colorado Springs, CO 80904
(303) 633-0211
Manager-Owner: David C. Cooper

Idaho

Electronic Specialties, Inc.
8411 Fairview Ave.
Boise, ID 83704
(208) 376-5040
Manager-Owner: Terry Romero

Wyoming

Computer Concepts
3125 E. Grand Ave. St.
Laramie, WY 82070

Utah

Computer Plus
1078 East Ft. Union Blvd.
Midvale, UT 84047
(801) 566-3902
Manager-Owner: Steve Whitzelor/
Allen Vincent

Mnemonics Memory Systems
(DBA Mnemonics Computer Store)
141 E. 200 South
Salt Lake City, UT 84111
(801) 266-7883
Manager: Rick Giolas

The Hi-Fi Shop
2236 Washington Blvd.
Ogden, Utah 84401
(801) 621-5244
Manager-Owner: Brent Richardson

Arizona

Personal Computer Place
1840 W. Southern Ave.
Mesa, AZ 85202
(602) 833-8949
Manager-Owner: Roger Smith

Computer Depot
1201 Iron Springs Rd.
Prescot, AZ 86301
(602) 778-7473
Manager-Owner: Brice Eldridge

Nevada

PCS Computer
3900 W. Charleston, Ste R
Las Vegas, NV 89102
(702) 870-4138
Manager-Owner: Mickey Cole

Harry's Business Machines, Inc.
323 West Street
Reno, NV 89501
(702) 322-4559
Manager-Owner: Gordon Foote

California

Opamp Tech Books
1033 N. Sycamore
Los Angeles, CA 90038
(213) 464-4322
Manager-Owner: Alicion

Opamp Tech Books
1033 N. Sycamore
Los Angeles, CA 90038
(213) 464-4322
Manager-Owner: Alicion

Dealers continued . . .

Data Equipment Supply Corp.
8315 Firestone Blvd.
Downey, CA 90241
(213) 923-9361
Manager: Robert Johnson

Computer Place
23914 Crenshaw Blvd.
Torrance, CA 90505
(213) 325-4754
Manager-Owner: Wen T. Huang

Fyrst Byte
10053 Whittwood Dr.
Whittier, CA 90603
(213) 947-9411
Manager-Owner: Darrell Miller

Game Room
5675 Kanan Room
Agora, CA 91301
(213) 707-0142
Manager-Owner: Jean Collier

General Computer Store
22323 Sherman Way Unit #7
Canogca Park, CA 91303
(213) 704-6600
Manager-Owner: Anita Broadway

HW Electronics
19511 Business Center Dr.
North Ridge, CA 91324
(213) 886-9200
Manager-Owner: Ronda

Levity Distributers
6857½ Ben Avenue
North Hollywood, CA 91605
(213) 982-2514
Manager-Owner: Melinda Plesha

Data Systems West
421 West Las Tunas Dr.
San Gabriel, CA 91776
(213) 289-3791
Owner: Frank J. Mogavero

Consumer Computers
8314 Parkway Dr.
La Mesa, CA 92041
(714) 465-8888
Manager: Steve Scott

Calco Digital Equpiment Inc.
1919 Aple St.
Oceanside, CA 92054
(714) 433-4119
Vice President: Ronald N. Paperno

20-64 Software Center
9829 Mira-Mesa Blvd
San Diego, CA 92131
(619) 695-0214
Manager-Owner: Larry Skaggs

Inland Electro Mart
8624 California Ave.
Riverside, CA 92504
(714) 687-3776
Manager-Owner: Jack

Quality Computer Center
801 S. Victoria St., #104
Ventura, CA 93003
(805) 642-1979
Manager-Owner: David Stewart

Jay-Kern Electronics
1135 Columbus
Bakersfield, CA 93305
871-5800
Manager-Owner: Don Taylor

Micro Pacific Computer Center
5148 N. Palm
Fresno, CA 93704
(209) 229-0101
Manager-owner: Mike Reinhold

J. Snell & Co., Inc.
657 Mission St.
San Francisco, CA 94105
(415) 421-5898
Manager-Owner: James Snell

PC Computers
10166 San Pablo Ave.
El Cerrito, CA 94503
(415) 527-6044
Manager-Owner: Gary Guttebo

Computer Ideas, Inc.
1029 Tennesee Street
Vallejo, CA 94590
(707) 552-5076
Manager-Owner: J. Gavin

Fox Computer Co.
2678 N. Main Street
Walnut Creek, CA 94596
(415) 944-9277
Manager-Owner: Stan Nielson

Albany Typewriter & Computer
923 San Pablo Ave.
Albany, CA 94706
(415) 526-1959
Manager-Owner: Bill Tichy

Idea Computers
301 North Santa Cruz Avenue
Los Cratos, CA 95030
(408) 354-1210
Manager-Owner: Tom Wolf

The Computer Room
230 Mt. Herman Rd.
Scotts Valley, CA 95066
(408) 438-5001
Manager-Owner: Gary Guttebo

The Computer Center Stores
930 Town & Country Village
San Jose, CA 95128
(408) 246-5710
Manager-Owner: R. Reid,
J. Barlow, N. Kinney

Inland Electro Mart
8624 California Dr.
Riverside, CA 95204
(714) 687-3776
Manager-Owner: Jack

Educational Connection
1508 Coffee Rd.
Modesto, CA 95355
(209) 576-1611

Software Plus
6201 "C" Greenback Lane
Citnes Heights, CA 95610
(916) 726-4979
Manager Owner: Carolyn Webster

The Radio Place
2964 Freeport Bl.
Sacramento, CA 95818
(916) 441-7388
Manager-Owner: Gary Stilwell

Ray Morgan Co.
554 Rio Lindo Ave.
Chico, CA 95926
(916) 343-6065
Manager: Dave Wegner

Computer Place
1698 Market St.
Redding, CA 96001
(916) 221-1312
Manager-Owner: John Fredricks

Radio Mart
1075 Cypress
Redding, CA 96001
(916) 241-3000
Manager-Owner: John Cokeley

Oregon

SW Computers
1125 N.E. 82nd
Portland, OR 97220
Manager-Owner: Jerry

Edu-Tech
1575 N.W. 9th

Corvallis, OR 97330
(503) 758-5577
Manager-Owner: L. Clark/W. Brown

Ace-Tec
Highway 101 S. Sypress Center

Bandon, OR 97411
(503) 347-9322
Manager-Owner: Ace Egnew

Washington

Compu-Play
1320 S. 324th Suite A-3
Federal Way, WA 98003
(206) 839-4453
Manager-Owner: Kevin Mitchell

Electronic Supermart
7040 S. 180th St.
Kent, WA 98032
(206) 251-8484
Manager-Owner: Richard Thorp

Programs Plus
16874 Southcenter Parkway
Seattle, WA 98188
(206) 575-1375
Manager-Owner: Nick Smith

Conti Electronics Ltd.
c/o Afcon
140-14th
Blaine, WA 98230
Manager-Owner: G.W. Harder

Computer Corner
1610 N. Laventure
Mt. Vernon, WA 98273
(206) 428-1840
Manager-Owner: Kirk D. Shroyer

Computer +
2504 Jefferson Ave
Tacoma, WA 98402
(206) 272-2329
Manager-Owner: David Dodd

Computer Town
1215 Center
Tacoma, WA 98409
(206) 272-2271

Nibles & Bytes
4020 S. Steel St. Suite 105
Tacoma, WA 98409
(206) 475-5938
Manager-Owner: John Clark

The Electronics Shop
131 N. Decator
Olympia, WA 98502
(206) 357-6304
Manager-Owner: Frank & Tim Linaham

Bits, Bytes and Nibbles, Inc.
209 Northtown Shopping Center
Spokane, WA 99207
(509) 487-1601
Manager-Owner: Richard Shulman

Alaska

BG Systems Co.
204 East International
Anchorage, AK 99502
(907) 276-2986
Manager-Owner: Robert DeLoach

Micro Age Computer Store
2440 Seward Highway
Anchorage, AK 99503
(907) 279-6688
Manager-Owner: Jay Wisthoff

CANADA

Systems Ornic Ltd
999 deBouragogue
Sinte Foy, Quebec G1W 4S6
Manager-Owner: Yvon Labbee

Caleq Inc.
331 Sir Walter Lourier Blvd.
St. Lambert, Quibec J4R 2L1
Manager-Owner: Marcel Bourcier

Electronics 2001
5529 Yonger Street
Willowdale, Ontario M2N 5S3
Manager-Owner: Chris Bennett

House of Computers
368 Eglinton Ave. W.
Toronto, ON M5N 1A2
(416) 482-4336
Manager-Owner: Mark Herzog

The Computer Circuit Ltd.
733 Richmond Street
London, Ontario N68 3H2

Kelley Software Dist. Ltd
P.O. Box 11932
Edmonton, Alberta T5J 3L1
Manager-Owner: Robert Owen

PROGRAM 3

```
** THIS IS CALLED WITH THE SYS 828 **
** IT PLACES A JMP $0359 IN THE    **
** CHRGET ROUTINE                  **
*
033C 094C   LDA #$4C OP CODE OF JMP
033E 857C   STA $7C
0340 A959   LDA #$59 MSB OF ADDRESS
0342 857D   STA $7D
0344 A903   LDA #$03 LSB OF ADDRESS
0346 857E   STA $7E
0348 A8     TAY

** $CB1E IS A ROM ROUTINE WHICH **
** WILL PRINT A MESSAGE FROM THE**
** ADDRESS SPECIFIED IN THE ACC **
** AND Y REGISTERS (Y=HIGH BYTE,**
** ACC=LOW BYTE)                **

034B 201ECB JSR $CB1E
034E 60     RTS
034F 0D     FCB $0D CARRIAGE RETURN
0350 444F53 FCC 'DOS OK.'
     204F4B
     2E
0357 0D     FCB $0D CARRIAGE RETURN
0358 00     FCB $00
0359 C95D   CMP #$5D  IS CHAR A ']'
035B D008   BNE $0365 NO
035D 48     PHP
035E A57B   LDA $7B
0360 C902   CMP #$02 IN DIRECT MODE?
0362 F009   BEQ $036D
0364 68     PLA
0365 C93A   CMP #':' IS IT A COLON?
0367 9001   BCC $036A
0369 60     RTS
036A 4C8000 JMP $0080 CONTIN CHRGET
036D 68     PLA
036E 207300 JSR $0073 GET FIRST LETTER
0371 9013   BCC $0386 SKIP IF NUMBER
0373 C94B   CMP #'K  IS FIRST LETTER A 'K'
0375 D014   BNE $038B BRANCH IF NOT
** KILL COMMAND RESTORES ORIGINAL **
** BYTES IN THE CHRGET ROUTINE    **
0377 A9C9   LDA #$C9
0379 857C   STA $7C
037B A93A   LDA #$3A
037D 857D   STA $7D
037F A9B0   LDA #$B0
0381 857E   STA $7E
0383 4C74C4 JMP $C474 JUMP TO BASIC READY
0386 A20B   LDX #11  ERROR 11 (SYNTAX)
0388 4C3AC4 JMP $C43A PRINT ERROR MESSAGE
038B 8500   STA $00
038D A900   LDA #$00
038F 20BDFF JSR $FFBD KERNAL 'SETNAM' ROUTINE
0392 A97F   LDA #$7F  LOGICAL FILE #127
0394 A208   LDX #8  DEVICE #8
0396 A00F   LDY #15  SECONDARY ADDR #15
0398 20BAFF JSR $FFBA SET UP LOGICAL FILE
039B 20C0FF JSR $FFC0 GO OPEN FILE
039E A500   LDA $00   GET BACK FIRST CHARACTER
03A0 C945   CMP #'E  IS IT 'E'?
03A2 F01C   BEQ $03C0 YES, GO OPEN FOR INPUT
03A4 A27F   LDX #$7F  NO, OPEN FOR OUTPUT
03A6 20C9FF JSR $FFC9 KERNEL 'CHKOUT' ROUTINE
03A9 A500   LDA $00   GET FIRST CHARACTER
03AB 20D2FF JSR $FFD2 PRINT TO DISK
03AE 207300 JSR $0073 GET NEXT CHARACTER
03B1 C900   CMP #$00  NULL CHARACTER?
03B3 D0F6   BNE $03AB NO, PRINT IT TO DISK
03B5 20CCFF JSR $FFCC CLEAR SERIAL BUS
03B8 A97F   LDA #$7F
03BA 20C3FF JSR $FFC3 CLOSE FILE #127
03BD 4C74C4 JMP $C474 GO PRINT 'READY'
03C0 A27F   LDX #$7F  OPEN FILE FOR INPUT
03C2 20C6FF JSR $FFC6 'CHKIN'
03C5 A90D   LDA #$0D  PRINT CARRIAGE RETURN
03C7 20D2FF JSR $FFD2
03CA 20CFFF JSR $FFCF GETS ERROR MESSAGE
03CD 20D2FF JSR $FFD2 PRINT TO SCREEN
03D0 C90D   CMP #$0D  CARRIAGE RETURN??
03D2 D0F6   BNE $03CA NO, GET NEXT CHARACTER
03D4 F0DF   BEQ $03B5 BRANCH TO 'READY'
03D6 00     BRK
03D7 00     BRK
03D8 00     BRK
```

SUBSCRIBE TO

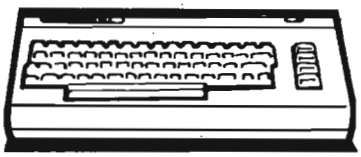
Commander

Now, and take advantage of our Charter Subscriber Discount of $4 OFF

THE MONTHLY JOURNAL FOR COMMODORE COMPUTER USERS

SUPER PET

VIC - 20



64

PET/CBM

"COMMANDER will be dedicated to communicating the fun of, as well as the latest information about the ***COMMODORE COMPUTERS.***"

EACH MONTH COMMANDER WILL HAVE:

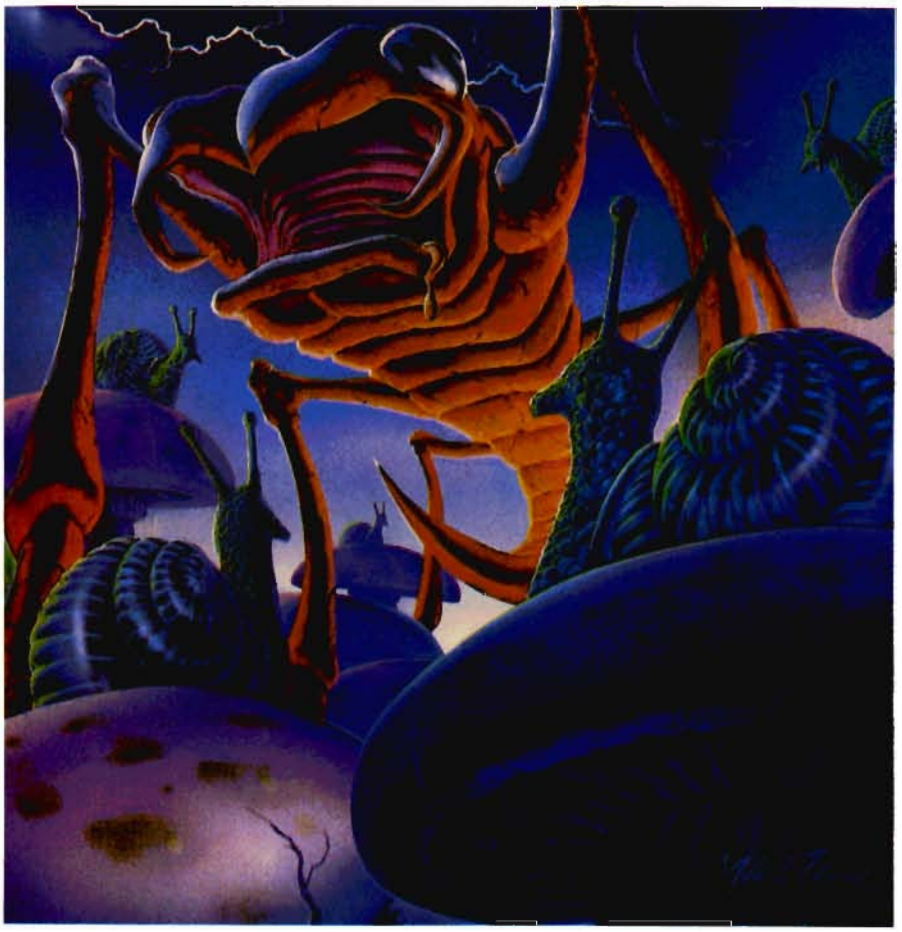
- the latest information and news releases
- software for education, business and fun
- reviews on hardware and software
- program listings
- application (how-to) articles
- a contest and MUCH, MUCH MORE!!

DON'T MISS OUT

on the most informative magazine dedicated to the COMMODORE COMPUTERS

HAVE YOU GOT WHAT IT TAKES TO BE A *Commander?*

Game—CONTEST



The Game Contest is a continuing feature of Commander magazine aimed at providing entertainment for and promoting competition among our readers. United Microware Industries has graciously provided us with February's Game Contest.

DEADLINE FOR ENTRIES: APRIL 1, 1983

Video Vermin may be purchased from United Microware Industries or any one of its fine dealers.

3503-C Temple Ave.
Pomona, California
1-714-594-1351

Terms for Game Contest

First prize will be awarded to the person with the highest score. The winning entry must contain a photograph of the highest score of the game. A Video Vermin package front and proof of purchase slip.

Entries must be mailed to Commander, Video Vermin Contest, PO Box 98827, Tacoma, WA 98498. All entries must be mailed, as postmarks are required to determine the earliest winning entry. In the event of a tie, duplicate prizes will be awarded. Employees of United Microware Industries and their families may not participate. First prize will be $250, second prize $100, third prize $50.

The contest will run until a first prize is awarded. United Microware Industries will notify Commander Magazine of the winner(s) and provide copy and photographs for a follow-up story.

Video Vermin

Your once-peaceful garden is being invaded by a hoard of hungry Vermin. Shoot them and their fallen bodies sprout a barricade of mushrooms to conceal more swarming raiders. Snails oil path on which the Vermin slither erratically toward you. Dropping butterflies leave a trail of mushrooms in their wake, and pouncing spiders, ants, beetles, and fleas infest the area. One vigilant gardener strikes first. If he's destroyed by the fatal touch of the invaders, two more defenders lie in wait to take his place. If the third gardener succumbs, and you have 10,000 points, another gardener will emerge to help save the garden from devastation.

I thought it was going to be a peaceful afternoon, just me and my

Vic. I turned my set on, and started to play Video Vermin, the Centipede type arcade game by UMI. Before I realized it I found myself in the midst of a large infestation of snails, butterflies, beetles, fleas and ants. I was trying diligently to kill them but to my dismay they just sprouted into mushrooms. It seemed as though my garden would never flourish.

Video Vermin is a fast paced intense arcade game. Your garden is being attacked by vermin, and as you shoot them they sprout into mushrooms which rain down on your garden. You begin the game with three gardners and with each 10,000 points you are awarded an extra man. The highest score is saved from game to game, and as you improve your aim the skill level increases automatically, so it's a constant battle to keep ahead.

Video Vermin is a fantastic Centipede type arcade game sold by UMI, with the highest arcade quality features. The sound is superb and the graphics are so excellent that I found myself reaching for the big spray more than once.

I recommend Video Vermin to anyone who loves fast action, great sound and realistic graphics. All you'll need to play this great game is a VIC-20, one joystick, a fast hand, and a cool head!

Debbie Gabbard

Sound Shaper

Continued from page 7

ferent types of filters can be selected. The filter resonance can also be changed.

The POKE commands necessary for the sound desired can be displayed at any time on the screen. The POKE commands can be printed out at any time to the VIC-1525 printer.

Sound Shaper for the Commodore 64

Cassette $9.95
Diskette $11.95

A Super Editor for the Pet

Continued from page 28

files to the disk for later assembly. PUT has the ability to output all or part of the memory resident file. The CPUT command will remove extra spaces not in comments or ASCII text.

LOADER

Syntax: LOADER "file name"(,hex offset)

The loader command does the same function as the separate loaders on the disk. The hex offset allows the object code to be stored at an address other than the assembled address. The same keys as for Basic-Aid can be used for to hold, pause, and stop the listing.

ATUG (ASM/TED Users Group)
c/o Brent Anderson
200 S. Century
Rantoul, Ill 61866
USA
217-893-4577

TPUG (Toronto PET Users Group)
c/o Chris Bennett
381 Lawrence Ave. West
Toronto, Ontario
Canada M5M 1B9
416-782-9252

Advertising Index

LUNA SOFTWARE

LUNA SOFTWARE is quickly becoming one of the largest manufacturers of software for the Commodore 64™ and Vic 20™. The reason behind our growth is that we offer the consumer only the best in quality, service and price. We have now available for immediate delivery a diverse line of software — ranging from business programs to arcade style games. Call us today and shoot for the stars with LUNA.

DISKETTES & CASSETTES FOR THE COMMODORE 64™ AND VIC 20™.

FIREFIGHTER 64
STRATEGY GAME
Protect your homes and forest from the raging fire in this exciting strategy game.
Available in Cassettes for CM/64.

GAME GRAPHICS EDITOR
UTILITY PROGRAM
A must for all serious programmers. It lets you make multi-colored sprites at the touch of a button.
Available in Cassettes for Vic 20.

MAILING LIST
HOME/BUSINESS PROGRAM
Alphabetizing, search on any field, over 650 records per diskette, and much more make this the most powerful mailing list program available.
Available in Diskettes for CM/64 and Cassettes for Vic 20

PERSONAL FINANCE
PERSONAL PROGRAM
This versatile program can also be used by small businesses as an expense register and general ledger.
Available in Diskettes for CM/64 and Vic 20.

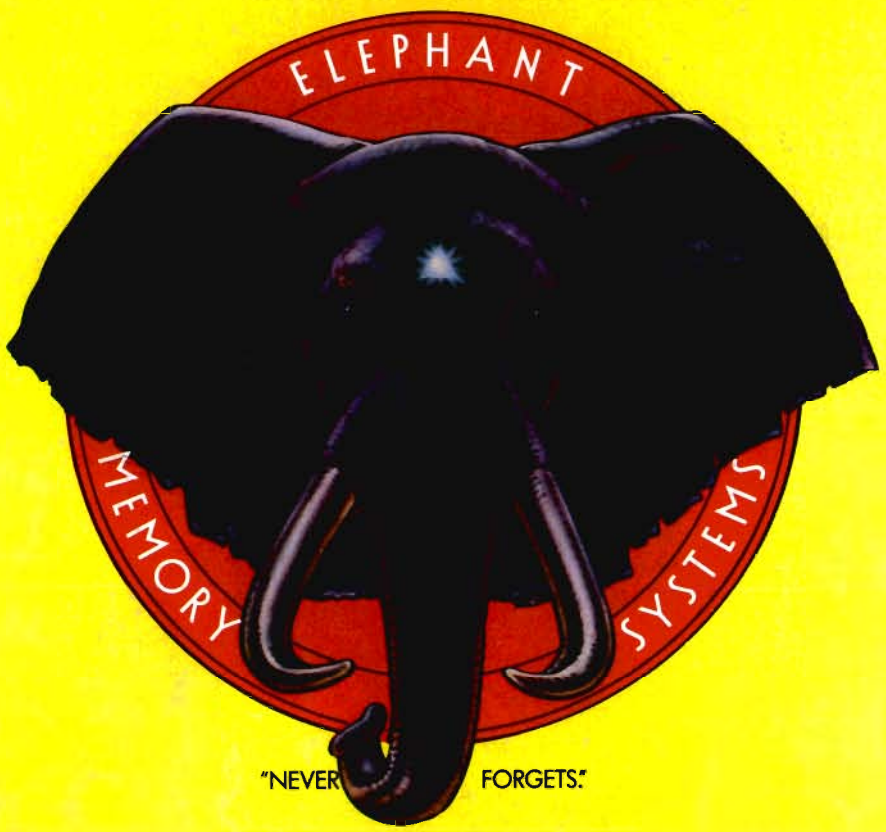
PEDESTRIAN
ARCADE STYLE GAME
Dart across busy boulevards and a perilous park as you avoid hazards and muggers in this fun and exciting game for all levels of players.
Available in Cassettes for CM/64.

...AND SO MANY MORE.
The above are only a few of the many programs that we have available.
To receive more information contact:

P.O. Box 26922
San Jose, CA 95159-6922
408-378-7793

LUNA SOFTWARE

REMEMBER:



MORE THAN JUST ANOTHER PRETTY FACE.

Says who? Says ANSI.

Specifically, subcommittee X3B8 of the American National Standards Institute (ANSI) says so. The fact is all Elephant™ floppies meet or exceed the specs required to meet or exceed all their standards.

But just who is "subcommittee X3B8" to issue such pronouncements?

They're a group of people representing a large, well-balanced cross section of disciplines—from academia, government agencies, and the computer industry. People from places like IBM, Hewlett-Packard, 3M, Lawrence Livermore Labs, The U.S. Department of Defense, Honeywell and The Association of Computer Programmers and Analysts. In short, it's a bunch of high-caliber nitpickers whose mission, it seems, in order to make better disks for consumers, is also to make life miserable for everyone in the disk-making business.

How? By gathering together periodically (often, one suspects, under the full moon) to concoct more and more rules to increase the quality of flexible disks. Their most recent rule book runs over 20 single-spaced pages—listing, and insisting upon—hundreds upon hundreds of standards a disk must meet in order to be blessed by ANSI. (And thereby be taken seriously by people who take disks seriously.)

In fact, if you'd like a copy of this formidable document, for free, just let us know and we'll send you one. Because once you know what it takes to make an Elephant for ANSI . . .

We think you'll want us to make some Elephants for you.

ELEPHANT.™ HEAVY DUTY DISKS.

For a free poster-size portrait of our powerful pachyderm, please write us.
Distributed Exclusively by Leading Edge Products, Inc., 225 Turnpike Street, Canton, Massachusetts 02021
Call: toll-free 1-800-343-6833; or in Massachusetts call collect (617) 828-8150. Telex 951-624.