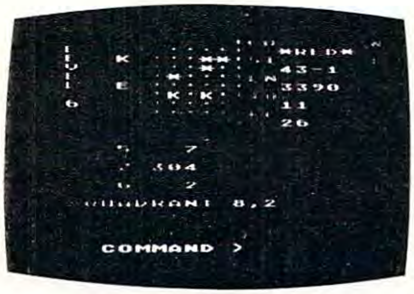
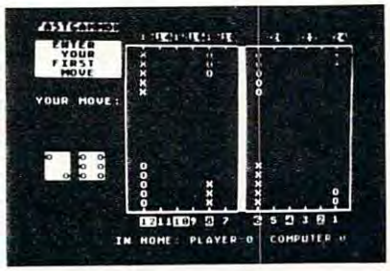
SOFTWARE FOR THE ATARI 800* AND THE ATARI 400*



TARI TREK™
By Fabio Ehrengruber

Get ready for an exciting trek through space. Your mission is to rid the galaxy of Klingon warships, and to accomplish this you must use strategy to guide the starship Enterprise around stars, through space storms, and amidst enemy fire. Sound and color enliven this action-packed version of the traditional trek game. Nine levels of play allow the player to make the mission as easy or as challenging as he wishes. At the highest level you are also playing against time. Damage to your ship can be repaired in space at a cost of time and resources if you can't make it back to base. TARI TREK gives you a lot of trek at a low price. This program is written entirely in BASIC and requires at least 24K of user memory. For the Atari 800 only.

Cassette - $11.95 Diskette - $14.95



FASTGAMMON™
By Bob Christiansen

Play backgammon against a talented computer opponent. This is the latest and best version of the most popular backgammon-playing program for personal computers - FASTGAMMON. Roll your own dice or let the computer roll them for you. Adjust the display speed to be fast or slow. If you wish you can play a game using the same dice rolls as the previous game - a great aid in improving your skills at backgammon. Beginners find it easy to learn backgammon by playing against the computer, and even very good players find it a challenge to beat FASTGAMMON. The 12-page instruction booklet includes the rules of the game. Written in machine language. Requires only 8K of RAM and runs on both the Atari 400 and the Atari 800.

On cassette only - $19.95

TANK TRAP
By Don Ursem

A rampaging tank tries to run you down. You are a combat engineer, building concrete barriers in an effort to contain the tank. Use either the keyboard or an Atari joystick to move your man and build walls. If you trap the tank you will be awarded a rank based on the amount of time and concrete you used up. But they'll be playing taps for you if you get run over. There are four levels of play. Higher levels of play introduce slow curing concrete, citizens to protect, and the ability of the tank to shoot through any wall unless you stay close by. Music, color, and sound effects add to the excitement. Written in BASIC with machine language subroutines. Requires at least 16K of user memory. Runs on the Atari 800 and on an Atari 400 with 16K RAM.

Cassette - $11.95 Diskette - $14.95

QS FORTH™ By James Albanese. Step into the world of the remarkable FORTH programming language. Writing programs in FORTH is much easier than writing them in assembly language, yet FORTH programs run almost as fast as machine code and many times faster than BASIC programs. QS FORTH is based on fig-FORTH, the popular model from the FORTH Interest Group that has become a standard for microcomputers. QS FORTH is a disk-based system that can be used with up to four disk drives. There are five modules included:

1. The FORTH KERNEL (The standard fig-FORTH model customized to run on the Atari computer).
2. An EXTENSION to the basic vocabulary that contains some handy additional words.
3. An EDITOR that allows editing source programs (screens) using Atari type editing.
4. An IOCB module that makes I/O operations easy to set up.
5. An ASSEMBLER that allows defining FORTH words as a series of 6502 assembly language instructions.

Modules 2-5 may not have to be loaded with the user's application program, allowing for some efficiencies in program overhead. Full error statements (not just numerical codes) are printed out, including most disk error statements. QS FORTH requires at least 24K of RAM and at least one disk drive. For the Atari 800 only.

On diskette only - $79.95

★ ★ ★ ★ ★ ★ ★ ★ ★ ★

ASSEMBLER by Gary Shannon. Write your own 6502 machine language programs with this inexpensive in-RAM editor/assembler. Use the editor to create and edit your assembler source code. Then use the assembler to translate the source code into machine language instructions and store the code in memory. Simple commands allow you to save and load the source code to and from cassette tape. You can also save any part of memory on tape and load it back into RAM at the same or at a different location. The assembler handles all 6502 mnemonics plus 12 pseudo-ops that include video and printer control. Commenting is allowed and error checking is performed. A very useful feature allows you to view and modify hexadecimal code anywhere in memory. Instructions on how to interface machine language subroutines to your BASIC programs are included. ASSEMBLER requires 16K of user memory and runs on both the Atari 800 and the Atari 400.

On cassette only - $24.95

★ ★ ★ ★ ★ ★ ★ ★ ★ ★

6502 DISASSEMBLER by Bob Pierce. This neat 8K BASIC program allows you to disassemble machine code, translating it and listing it in assembly language format on the video and on the printer if you have one. 6502 DISASSEMBLER can be used to disassemble the operating system ROM, the BASIC cartridge, and machine language programs located anywhere in RAM except where the DISASSEMBLER itself resides. (Most Atari cartridges are protected and cannot be disassembled using this disassembler.) Also works as an ASCII interpreter, translating machine code into ASCII characters. 6502 DISASSEMBLER requires only 8K of user memory and runs on both the Atari 800 and the Atari 400.

Cassette - $11.95 Diskette - $14.95

QS QUALITY SOFTWARE
6660 Reseda Blvd., Suite 105, Reseda, CA 91335
(213) 344-6599

WHERE TO GET IT: Call us at (213) 344-6599 for the name of the Quality Software dealer nearest you. If necessary you may order directly from us. Mastercard and Visa cardholders may place orders by telephone. Or mail your check or bankcard number to Quality Software, 6660 Reseda Blvd., Suite 105, Reseda, CA 91335. California residents add 6% sales tax. SHIPPING CHARGES: Within North America orders must include $1.50 for first class shipping and handling. Outside North America the charge for airmail shipping and handling is $5.00. Pay in U.S. currency.

```
300 POKE J+148,24
310 POKE J+149,126
320 POKE J+150,171
330 POKE J+151,126
340 POKE J+152,24
350 POKE 704,200
360 POKE 53248,70
```

Place photon cannon image data into second player's mapping buffer area and set color to orange. Set cannon's initial horizontal position to 160.

```
400 POKE J+347,24
410 POKE J+348,60
420 POKE J+349,24
430 POKE J+350,24
440 POKE J+351,60
450 POKE J+352,126
460 POKE J+353,255
470 POKE J+354,102
480 POKE 705,56
490 CNPOS=160
```

Set saucer player to double width and enable P/M graphics.

```
500 POKE 53256,1
510 POKE 53277,3
```

Initialize game counters and registers.

```
600 HIGH=0
700 M=0:SCORE=0:CN=1
```

This ends program initialization. Begin process loop. Determine saucer's position.

```
1000 FOR POS=15 TO 240 STEP 5
1010 SCPOS=POS
```

Vary saucer position if in range of cannon.

```
1020 IF ABS(POS-CNPOS)<20 AND M<>0 THEN
SCPOS=POS+(25-50*RND(0))
1030 POKE 53248,SCPOS
```

Produce saucer sound effect.

```
1040 FRQ=FRQ+1:IF FRQ>6 THEN FRQ=1
1050 SOUND 0,60/FRQ,10,5
```

If missile is not launched check if in range to activate saucer death ray.

```
1060 IF M<J+12 THEN IF ABS(CNPOS-SCPOS)<
15 THEN 3100
```

If missile is not launched or has reached the top of the screen, set missile flag and sound to zero. Check fire button, if pressed init missile position at cannon.

```
1100 IF M<J+8 THEN M=0:SOUND 1,0,0,0:IF
STRIG(0)=0 THEN M=J+93:POKE 53253,CNPOS+
3
```

If missile is launched then move it to next vertical position.

```
1110 IF M>0 THEN POKE M,0:POKE M-1,0:M=M
-8:POKE M,12:POKE M-1,12:SOUND 1,ABS(M+5
0-J)*0.5,8,15
```

Check missile collision register. If hit then goto 3000 for effect.

```
1120 IF PEEK(53257)=1 THEN 3000
```

Determine cannon horizontal position.

```
1200 IF STICK(0)=11 THEN IF CNPOS>50 THE
N CNPOS=CNPOS-5
1210 IF STICK(0)=7 THEN IF CNPOS<198 THE
N CNPOS=CNPOS+5
```

```
1220 POKE 53249,CNPOS
```
End process loop.
```
2000 NEXT POS:GOTO 1000
```
Saucer hit and explosion handler erase missile, change saucer color, and update earth score.
```
3000 FOR X=0 TO 28:POKE X+J,0:NEXT X:M=0
3010 POKE 704,INT(15*RND(0))*16+8
3020 SCORE=SCORE+25
```
Set explosion in sky.
```
3030 BK=4
3040 GOTO 4000
```
Saucer death ray handler. Position death ray position under saucer.
```
3100 POKE 53252,SCPOS+7
```
Fire ray and produce sound.
```
3120 SOUND 1,4,12,15:SOUND 0,0,0,0
3130 FOR X=25 TO 97 STEP 3:POKE J+X,3:NE
XT X
3140 FOR X=25 TO 97 STEP 3:POKE J+X,0:SO
UND 1,X-23,12,15:NEXT X
3150 SOUND 1,0,0,0
```
Check cannon's missile collision register, if a hit then update cannon counter and produce explosion.
```
3200 IF PEEK(53256)<>2 THEN 2000
3210 CN=CN+1:POKE 53249,0
3220 BK=2
```
Explosion handler program.
clear collision registers and disable missile sound.
```
4000 POKE 53278,255
4010 SOUND 1,0,0,0
```
Produce explosion sound and light.
```
4020 FOR X=15 TO 0 STEP -1
4030 SETCOLOR BK,3,X
4040 SOUND 0,50,0,X
4050 FOR D=0 TO 15:NEXT D
4060 NEXT X
```
Restore colors for sky and earth.
```
4100 SETCOLOR 2,10,6
4110 SETCOLOR 4,0,0
```
Display score.
```
4200 POKE 84,0:POKE 85,2
4210 POKE 53248,0
4220 IF BK=2 AND CN<4 THEN ? #6;"CANNON
";CN:GOTO 4240
4230 ? #6;"SCORE     ";:POKE 85,8:? #6;S
CORE
4240 FOR D=0 TO 200:NEXT D
```
Check if end of game.
```
4300 IF CN<4 THEN 4400
4310 IF SCORE>HIGH THEN HIGH=SCORE
4320 POKE 84,0:POKE 85,2
4330 ? #6;"HIGH     ";:POKE 85,7:? #6;HIGH
4340 POKE 53279,8:FOR D=0 TO 80
4350 IF PEEK(53279)=6 THEN 700
4360 NEXT D:GOTO 4200
```
Restore background.
```
4400 POKE 84,0:POKE 85,2:? #6;"   .",".
   "
4410 GOTO 1000
```
©

Unleash The Power Of Your Atari CPU

Ed Stewart
Uniontown, OH

Would you like to get as much as a 30% increase in speed from your Atari 6502 CPU? Would you also like to get this benefit without any additional capital expense? If your answer is no you probably don't like apple pie either, but if you answer yes read on and I will tell you how to accomplish such a feat with one simple BASIC POKE in the right place.

First a little background information about one of the many things that is going on inside your Atari computer. The particular thing that I want you to know something about is how display information reaches your TV screen. There is a specific hardware chip called ANTIC that has most of the responsibility for seeing that the display gets to your TV screen. ANTIC does this by operating independently from the main 6502 CPU in your computer. ANTIC is in fact a primitive CPU in it's own right. It executes a program which is located in RAM just as the 6502 executes a program in RAM or ROM. We can therefore call the Atari a multiprocessing computer since more than one CPU may be active at any time. A peculiar and somewhat unfortunate thing happens when a multiprocessing system such as the Atari is actively executing instructions—both CPUs desire access to memory simultaneously. The two CPUs cannot both access memory at the same time so one must wait until the other completes it's access request. This memory access conflict is common to all computers containing more than one CPU—from micros to macros—and is generally not something to be concerned about.

The ANTIC chip fetches it's data from memory using a technique called "Direct Memory Access" or DMA. Whenever this memory fetch is occuring the 6502 is temporarily halted. DMA is said to be "stealing" a portion of the computer's available time called a cycle. There are 1,789,790 cycles of computer time available per second. If DMA had not "stolen" that cycle of computer time the 6502 would not have been halted and therefore would have finished it's program instructions sooner. It is only logical to conclude that the more this DMA activity occurs in behalf of the ANTIC chip, the more our 6502 will be slowed down.

The ANTIC chip re-displays the entire TV display 60 times each second. During each of these 60 times many computer cycles are stolen from the 6502. During each of these 60 times the ANTIC chip also "interrupts" the 6502 and causes it to perform such tasks as updating various software timers and reading game controllers (joysticks and paddles). When the 6502 finishes what it must do in response to the ANTIC "interrupt" it may continue with what it was doing previous to being sidetracked by ANTIC. You should be getting the picture by now that although ANTIC is indispensable it causes a slowdown in the 6502 CPU, but how much?

I wrote a simple BASIC program for my Atari 800 in an attempt to answer this question. A FOR/NEXT loop was executed 100,000 times with no intervening statements as follows:

```
20 FOR I = 1 TO 100000:NEXT I
```

The first thing to measure was how long this loop executes with no ANTIC DMA active. A POKE 559,0 turned DMA off and the TV screen went black. A POKE 559,34 turned DMA back on and the original display was restored. The FOR/NEXT loop with *no* DMA required 148 seconds to complete. This same FOR/NEXT loop was executed in graphics modes 0-8 *with* DMA active and the execution times were observed as shown in table 1. The execution times *with* DMA increased from as little as 10% for graphics 3 to as much as 47% for graphics 8. It is reasonable to see that if you do a lot of number crunching and you don't need the TV screen or the software timers and game controllers then turn off the antic DMA for a while and you'll get your answer back sooner. It is also apparent from table 1 that your programs will execute faster if you are executing in graphics modes 3,4, or 5.

I hope you have learned a little bit more about the Atari computer and how the ANTIC DMA interferes with the 6502 CPU. You may in fact someday be able to unleash that latent power within during a computer chess tournament, and when someone asks how in the world you did it you can smile and say -me and my DMA.

GRAPHICS MODE	EXECUTION SECONDS	% INCREASE (over no-DMA)
NO DMA	148	
GRAPHICS 0	216	46
GRAPHICS 1	188	27
GRAPHICS 2	186	26
GRAPHICS 3	163	10
GRAPHICS 4	164	11
GRAPHICS 5	167	13
GRAPHICS 6	173	17
GRAPHICS 7	185	25
GRAPHICS 8	218	47

Table 1: Execution timings with and without DMA active for various graphics modes. ©

String Arrays in Atari BASIC

Charles Brannon

This article describes a method to simulate string array handling in Atari BASIC. If you already know what a string array is, skip ahead a bit. Otherwise, read on...

An array is essentially a list. For example, the numeric array A could hold a list of monthly amounts. A(1) would hold January, A(2) would contain February's monthly total, and so on, A(12) containing December's amount. To print the amounts from month one to month twelve, the BASIC statement:

```
FOR I = 1 TO 12:PRINT A(I):NEXT I
```

could be used. As the amounts were printed, the computer would optionally add the amounts up and print a yearly total. Numerical arrays also have many uses in mathematics.

If we wanted to print the name of each month along with the amount for that month, string arrays would come in handy. For January, the string array M$ might hold "Jan" (i.e. M$(1) = "JAN"). Once all the strings were defined, we could print the months along with the amounts:

```
FOR I = 1 TO 12:PRINT M$(I), A(I):NEXT I
```

Alternately, a string array could contain a list of player's names in a game. **PRINT P$(PLR)** could give the name of player number PLR.

It is a fairly well-known fact that Atari BASIC is not Microsoft BASIC (although a Microsoft BASIC for Atari should be released soon). The designers of the BASIC decided to use a different way of manipulating strings than Microsoft did. The statement **PRINT LEFT$(A$,4)** in Microsoft BASIC would produce the same effect as **PRINT A$(1,4)** would in Atari BASIC. Unfortunately, this notation precludes the use of string array notation.

Nevertheless, it is possible to produce the same effect as string arrays under Atari BASIC. The technique used here is similar to the one used in the string sort program in the back of the *Atari Basic Reference Manual.*

Essentially, the solution is to partition one very large string into several substrings. Each substring will be an element of the larger array. First, let us review the format that Atari BASIC uses to handle substrings. Say that A$ contains "Hello, how are you?". To print the word "how" we would specify A$(8, 10), the first argument, eight, being the starting position within the string and the second being the ending position of the substring, in this case, 11. We use this notation to recall or store any element from the main string. There are some non-Microsoft notations to specify an element in a string array. **PRINT A$(2,3,6)** would print the characters from three to six in the second element of A$. We will use an ''unraveled'' string to simulate this.

Say we want to set aside an area of memory for 10 strings, each string having a maximum length of 20 characters. Actually, this is required in Atari BASIC anyway. Since the DIM statement only takes one argument, the total number of characters in the string, we would enter in the program the statement: DIM A$(200), as 10 strings of 20 characters each would be 200 characters in total length. Now, to define each string, we must specify the starting location within the main string and how many characters to store at that location. If the string to be stored was T$, and E is the element that T$ is, then the statement to store T$ in A$ would be:

```
A$( (E-1)*20 + 1, E*20 ) = T$
```

If T = 2 (i.e. T$ would be the second string in A$), then the statement would reduce to: A$(21,40) = T$, and indeed, the area of A$ reserved for element 2 would be characters twenty-one to forty.

To recall an element from the string array, the statement need only be reversed. The element position of T$ still needs to be specified.

```
T$ = A$(E - 1)*20,E*20)
```

The ''20'' in the statements is of course the maximum number of characters in any element in the array. It could be specified a a variable at the start of your program.

There are still a few problems with this scheme. Although both CLR and RUN clear out the simple variables, the arrays and strings are left untouched. Therefore, before one can store a new element in the main string, a possible previous one must be cleared out. If the two previous statements were renumbered so as to be subroutines, then T$ could be set equal to twenty spaces first, GOSUB the storage routine, then store the actual value of the element. The example program should demonstrate this.

When printing the string, any character less than twenty characters will have trailing spaces. This could be corrected by a routine that strips off these spaces, but it might have a hard time if multiple spaces could be present in an actual element. Therefore, it might be advantageous to keep track of the length of each element so that only the proper number of characters would be printed. The length of each element in the array could be stored in a numeric array, say L. Therefore, the mini-subroutine to store a string element would be:

```
20000 L = LEN(T$): IF L>MAXLEN THEN L = MAXLEN
20010 L(E) = L : START = (E - 1)*MAXLEN
20020 A$(START, START + L - 1) = T$: RETURN
```

This subroutine requires that A$ and L be properly DIMensioned at the start of the program, MAXLEN equals the maximum length of each element, T$ contains the string to be stored, and E contains the number of which element T$ will become.

To recall an element, set E equal to the element number, and call the following routine. MAXLEN must be predefined (see above) and the strings and array L DIMensioned.

```
3000 START = (E - 1)*MAXLEN:T$ = A$(START,START +
  L(E) - 1):RETURN
```

To sum up, I have presented here a technique to simulate the use of string arrays. Undoubtedly it is not the only way, so be creative! Nevertheless, the two subroutines can be the foundation for programs of increased sophistication, as string arrays add a new dimension to programming efforts! ©

Atari Super Breakout

Robert Baker

If you own an Atari 800 computer don't forget there are other game cartridges besides STAR RAIDERS available. My favorite game (next to STAR RAIDERS of course!) is SUPER BREAKOUT. If you've visited any game arcades recently you may have seen an Atari Super Breakout machine. Well the SUPER BREAKOUT cartridge provides the same arcade games for your Atari 800 as well as standard BREAKOUT.

The cartridge provides four games: regular BREAKOUT, Progressive, Double, and Cavity. Each game can be played by one to eight players if you have enough paddle controllers. As expected, the major idea is to knock bricks out of a wall and score points determined by the brick colors.

Regular BREAKOUT is just like the Breakout game contained in the familiar Atari Pinball Breakaway TV game. You try to knock out all the bricks from 8 rows in the wall with five balls. If you knock out all the bricks, a new wall of bricks will appear. There is no limit on the number of times a new wall of bricks can be reset during a game.

With **Progressive**, the setup is somewhat different from regular BREAKOUT. When the game begins there are four rows of bricks at the top of the screen, followed by four blank rows, and then four rows of bricks. After game play begins, the brick walls move down toward the bottom of the screen at a rate determined by the number of times the ball is hit. At the same time, new rows of bricks enter the top of the screen at a progressively faster rate. This game will continue forever if you're good enough!

The **Double playfield** is the same as for regular Breakout except that there are two paddles and two balls are served. The paddles are stacked one on top of the other. The point values are the same as for other games except that each brick is worth double the normal amount whenever two balls are in play. In this game, the wall can only be reset twice after the initial wall is knocked out.

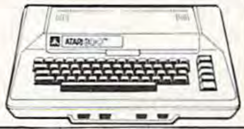
For **Cavity,** the playfield is just like regular Breakout except there are two "cavities" and each contains a ball. When the game begins, the balls bounce inside each cavity but are held captive. When enough bricks are removed to release a captive ball, that ball then enters play and starts to knock out bricks and score points. Point values are double with two balls in play or triple with three balls in play. You do get two paddles, as in Double, and the wall will only be reset twice.

There are no options to vary the number of balls or the paddle size like on the Pinball-Breakaway TV game. However, there is an option to suspend play and another to obtain an additional five serves if desired. After each game, scores are rated from "OOPS" to "BEST". Watch out—these games can be almost as addictive as STAR RAIDERS. They're also great party games since more than one person can play.

More Games For The Atari 400/800

Once you have an Atari 410 cassette drive for the Atari computer systems, there are a number of programs available from various sources. If you're interested in games then you might want to take a look at Mountain Shoot by **Adventure International.** This is a two player game where players take turns shooting cannons at each other. You enter the angle (0 to 90 degrees) to position the gun and the amount of powder (4 to 10) to be used. You even have to watch the wind direction and velocity that is always changing. Each time the game is run, a different terrain is generated between the two gun positions to make things even more challenging.

There are three levels of play: easy, hard, and very hard. The harder the level, the more accurate your shot must be to score a hit. In each level you can play with or without a powder limit. When playing



without limits, the game ends when the first player hits his opponent. With a powder limit, you continue playing until *both* players have exhausted their powder. The first player to use up all their powder has to sit by and watch their opponent fire away at him. The player with the most hits, when both players run out of powder, is the winner.

When loading from tape, you first load a brief set of instructions. These remain on the television screen while loading the actual game program. While the program is loading don't forget to turn up the sound on the television. Otherwise you'll miss additional loading precautions and game instructions that are given audibly while the 1812 Overture plays in the background. The program takes full advantage of the two channel cassette drive of the Atari system. It sure beats sitting around doing nothing waiting for a program to load!

There are no printed instructions provided with the game but once played, they'd be unnecessary anyway. The graphics used by the game are not overly attractive. However, you do get sound effects and even a fanfare when you score a hit. The game does not check for a correct angle being entered; it will accept any positive number. If you're not careful you can destroy your own gun and score an additional hit for your opponent. On the other hand, the amount of powder is limited to 10 maximum even if you enter a larger number. All in all, this is a fairly good two player game for the Atari and it's easy enough for young children to play. It's not as elaborate as some of the games from Atari but it can be fun to play.

Thesis (P.O. Box 147, Garden City, MI 48135) offers a number of programs on tape for the Atari computers at $15 each. Their Casino I tape contains two games: BLACKJACK and SLOT MACHINE. Brief printed documentation is included, sufficient for these type of games.

The BLACKJACK game accommodates one to four players and uses full casino rules. It allows you to hit or stand, split pairs, or even double down. The game provides unique graphics that show the dealer as he shuffles and deals the cards each time the deck is exhausted. However, cards are only shown as a face value and a suit symbol. No graphics are used for the cards.

The game does check betting limits, but allows any fraction or full floating point number ($1.23456) to be entered as a bet. If a value entered has a number of digits past the decimal point, it does not get cleared right away and messes up the display slightly. Eventually, however, everything does get corrected. I guess if you enter a value less than a cent, then you deserve to get strange results.

BLACKJACK is a 16K program, so it does take a little while to load from tape. Playing the game is pretty straight-forward. You only enter single letter responses to select your desired action. What else can I say, It's a standard Blackjack game with reasonable graphics and sound.

The SLOT MACHINE game, that comes with BLACKJACK, shows a graphic representation of a slot machine (with parts of other machines on either side). The yellow START button on the ATARI is used to pull and release the handle of the slot machine. Some kind of "noise" is heard as the "wheels" spin and the game makes any appropriate pay-offs depending on the odds. On larger wins, you even get to see coins come out of a slot at the bottom of the slot machine. Since only three symbols (cherry, lemon, & gold bar) are used on the "wheels", there seem to be a fair number of pay offs or wins. This program is not quite as good as some I've seen for other machines but it is still interesting.

The CRIBBAGE game from THESIS is a 24K program and does take a while to load from tape. The game is a standard game of Cribbage, you against the computer. Little graphics are used other than for the cribbage peg board used for scoring. Cards are shown only as a face value and a suit symbol. Printed documentation is provided and includes brief rules on cribbage, as well as directions for using the program and how it works. The game offers two levels of play: beginner and intermediate. The beginner level does not penalize for any errors during play or scoring while the intermediate level will. However, the computer plays the same strategy at both levels of play.

If you've played cribbage, then you know the various steps in each hand during the game. In part of each hand, you alternate playing cards and scoring points, then later score points on your hand for various combinations (runs, flushes, pairs, etc.). In this computerized version, you must specify the card to be played, the current card count, and any score that you claim for your card played. The computer doesn't do any of the work for you. Again, when you later score your hand you also have to enter your own claimed score. Part of this is because of the intermediate level of play where you can be penalized for missing a possible score.

This game could be very good except for one major disadvantage. The author chose to use a very poor method of selecting the card suit when the user must specify his card. Instead of using an easy to remember key letter (like C for Clubs, H for Hearts, etc.) you must enter the actual graphic symbol for the suit. Thus, you have to remember that Control P is used for Clubs, Control comma is Hearts, Control period is Diamonds, and Control semicolon is Spades. Of course none of these are marked on the keyboard so you have to memorize them or have a reference card handy. Otherwise the game is rather good but slow playing. ©

Speeding Up The Player-Missile Demo

Larry Isaacs, Raleigh, NC

Chris Crawford's article[1] on the inner workings of some of the player-missile graphics was very interesting. I'm sure all those who tried the demo program noticed the difference in speed between horizontal movement and vertical movement. This provides a very good example of the difference between the execution speed of machine language and the execution speed of BASIC.

Horizontal movement of the "player" requires only a POKE statement. The function of the POKE statement is roughly equivalent to a single machine language instruction. This allows it to execute fairly fast. On the other hand, vertical movement of the "player" isn't nearly as simple. A FOR...NEXT statement is needed to move some data in memory. Also, vertical movement requires a more complex POKE statement. This POKE statement not only takes longer to execute, but is executed 7 times. These factors result in noticeably slower vertical movement than horizontal movement. If this FOR...NEXT loop could be replaced with some machine language, vertical movement could be brought to seemingly the same speed as horizontal movement.

The program in Listing 1 will illustrate this point. This program is a duplication of Chris Crawford's original demo with modifications for upward movement of the "player" to be done with the aid of machine language. Downward movement is still done with the FOR...NEXT loop to give a comparison. With this program you will find that it takes around three seconds to move the "player" from the bottom of the screen to the top. It will take around 17 seconds to move the "player" back to the bottom again. If the downward movement is too slow to bear, use the program in Listing 2. It has machine language for both upward and downward movement.

The machine language routines do not contain any absolute addressing, so they are relocateable. This means you can further modify the demo programs and the routines will still work. For those familiar with assembly language, here is the code for the two routines.

```
UP      PLA
        PLA
        STA $CC
        PLA
        STA $CB
        LDY #$01
UPLOOP  LDA ($CB),Y
        DEY
        STA ($CB),Y
        INY
        INY
        CPY #$07
        BNE UPLOOP
        RTS
```

```
DOWN    PLA
        PLA
        STA $CC
        PLA
        STA $CB
        LDY #$06
DNLOOP  LDA ($CB),Y
        INY
        STA ($CB),Y
        DEY
        DEY
        CPY #$FF
        BNE DNLOOP
        RTS
```

As illustrated by this example, where the speed of an operation is concerned, it is faster to use machine language. However, it may not always be better to use machine language, and using it probably won't be easier than using BASIC.

Listing 1

```
1 GOSUB 1000:REM Load machine code
10 SETCOLOR 2,0,0:X=120:Y=48:REM Set bac
kground color and player position
20 A=PEEK(106)-8:POKE 54279,A:PMBASE=256
*A:REM Set player-missile address
30 POKE 559,46:POKE 53277,3:REM Enable P
M graphics with 2-line resolution
40 POKE 53248,X:REM Set horizontal posit
ion
50 FOR I=PMBASE+512 TO PMBASE+640:POKE I
,0:NEXT I:REM Clear out player first
60 POKE 704,216:REM Set color to green
70 FOR I=PMBASE+512+Y TO PMBASE+516+Y:RE
AD A:POKE I,A:NEXT I:REM Draw player
```

```
80 DATA 153,189,255,189,153
90 REM Now comes the motion routine
100 A=STICK(0):IF A=15 THEN GOTO 100
110 IF A=11 THEN X=X-1:POKE 53248,X
120 IF A=7 THEN X=X+1:POKE 53248,X
130 IF A=13 THEN FOR I=6 TO 0 STEP -1:PO
KE PMBASE+512+Y+I,PEEK(PMBASE+511+Y+I):N
EXT I:Y=Y+1
140 IF A=14 THEN D=USR(UP,PMBASE+511+Y):
Y=Y-1
150 GOTO 100
1000 DIM UPCODE$(22):UP=ADR(UPCODE$)
1010 FOR I=1 TO 5:READ A:NEXT I:REM Skip
 player data
1020 FOR I=UP TO UP+20
1030 READ BYTE:POKE I,BYTE
1040 NEXT I:RESTORE :RETURN
1050 REM Move player up code
1060 DATA 104,104,133,204,104,133,203
1070 DATA 160,1,177,203,136,145,203
1080 DATA 200,200,192,7,208,245,96
```

Listing 2

```
1 GOSUB 1000:GOSUB 1100:REM Load machine
 code
10 SETCOLOR 2,0,0:X=120:Y=48:REM Set bac
kground color and player position
20 A=PEEK(106)-8:POKE 54279,A:PMBASE=256
*A:REM Set player-missile address
30 POKE 559,46:POKE 53277,3:REM Enable P
M graphics with 2-line resolution
40 POKE 53248,X:REM Set horizontal posit
ion
50 FOR I=PMBASE+512 TO PMBASE+640:POKE I
,0:NEXT I:REM Clear out player first
60 POKE 704,216:REM Set color to green
70 FOR I=PMBASE+512+Y TO PMBASE+516+Y:RE
AD A:POKE I,A:NEXT I:REM Draw player
80 DATA 153,189,255,189,153
90 REM Now comes the motion routine
100 A=STICK(0):IF A=15 THEN GOTO 100
110 IF A=11 THEN X=X-1:POKE 53248,X:GOTO
 100
120 IF A=7 THEN X=X+1:POKE 53248,X:GOTO
100
130 IF A=13 THEN D=USR(DOWN,PMBASE+511+Y
):Y=Y+1:GOTO 100
140 IF A=14 THEN D=USR(UP,PMBASE+511+Y):
Y=Y-1
150 GOTO 100
1000 DIM UPCODE$(22):UP=ADR(UPCODE$)
1010 FOR I=1 TO 5:READ A:NEXT I:REM Skip
 player data
1020 FOR I=UP TO UP+20
1030 READ BYTE:POKE I,BYTE
1040 NEXT I:RETURN
1050 REM Move player up code
1060 DATA 104,104,133,204,104,133,203
1070 DATA 160,1,177,203,136,145,203
1080 DATA 200,200,192,7,208,245,96
1100 DIM DOWNCODE$(22):DOWN=ADR(DOWNCODE
$)
1120 FOR I=DOWN TO DOWN+20
1130 READ BYTE:POKE I,BYTE
1140 NEXT I:RESTORE :RETURN
1150 REM Move player down code
1160 DATA 104,104,133,204,104,133,203
1170 DATA 160,6,177,203,200,145,203
1180 DATA 136,136,192,255,208,245,96
```

©

Now Take Along Your Atari* in MICROCASE

lightweight, rugged protection for your Atari* investment

Perfect for Home, School, Office
Protects your Atari* from dirt, scuffs, damage
Made from durable CODURA
Completely machine washable
Holds Atari* 800 or 400 and accessories
Insulated with 5" thick high density foam
Measures: 14" x 18" x 10"
Weighs only 3 lbs. 10 oz.
Choose: Cobalt Blue with navy trim or Desert Tan with chocolate trim
Also available Microcase II for 810 Disk Drive, Atari* Printer, Modem, Modem Connector, two power packs

Soon:
Microcase Protection for other leading microcomputers.

Casemaker
13715 Quito Road
Saratoga, CA 95070

For ordering information call toll-free: **800-543-3000** Ext. **3622**

Available at leading computer stores including:
Anderson Computers
Alabama and Tennessee
Computer Corner
Ohio
Convenience Corner Electronics
Oklahoma
Math Box
Maryland, Virginia and Washington, D.C.
On-Line Micro Centers
California, Texas, Oklahoma, and Idaho
P.C.S. Computer Service
Nevada
Quality Technology
Utah
Video Vision Store
Tennessee

*A trademark of Atari used under license Indicates trademark of ATARI Inc.

SWIFTY SOFTWARE, INC.

SOFTWARE FOR ATARI

HOME MANAGEMENT

FILEIT by Jerry White. A Database System. Use this no-nonsense electronic cataloguing system to create, add, delete, edit, print, store and sort different kinds of custom files. For every need around your home. Disk Package runs on 24K and up. **$29.95** disk only.

PROGRAMMING AIDS

BASIC RENUMBER by Fernando Herrera. Change lines, references, even variables!! Includes **BCD** converter and BASIC PROGRAM DECODER. **$14.95** cassette; **$19.95** disk.

CHALLENGE & FUN

SPACE CHASE by Fernando Herrera. Conquer the Galaxy... if you can escape the deadly chase of the patrol fleet. 4 levels.
TIME BOMB by Fernando Herrera. Beat the clock in a Mine Field. 5 levels — any number of players — scoreboard lists top players.
MINDTRACE by Lee Jacknow. If you like Simon you will love this challenge...a real test for your memory.
DOTS-N-BOXES by Lee Jacknow. You played it as a kid...it's better on a computer.

Games on cassette; **$14.95** each
Deduct 20% on orders of 3 or more games
Add $5 to total if requesting disk

EDUCATIONAL

MY FIRST ALPHABET by Fernando Herrera. Before you send your kid to school he or she will know all the letters, numbers, musical tunes, animals...even typing and how to run a program!! Program allows parents to use built-in drawings or to design their own. **$29.95** disk only.

TO ORDER SEND CHECK OR MONEY ORDER TO:
SWIFTY SOFTWARE, INC.
P.O. BOX 641
MELVILLE, N.Y. 11747

Atari is a trademark of Atari, Inc. • N.Y. RESIDENTS ADD 7% SALES TAX

NEW!

ATARI* SOFTWARE

*Atari is a trademark of Atari, Inc.

- ***Highest Quality***
- ***Designed to be extremely user-friendly***
- ***Built-in instructions***
- ***Extensive error trapping***
- ***Auto prompting wherever is necessary***

The Gates of Adventure *- our fast-moving, science-fantasy game requiring the player to make heavy use of his memory. A journey through mazes of gates & planets! "Gates" makes use of the Atari's sound & color graphics capacity. The user can select a version which is compatible with the amount of RAM in his machine.*

TYPE	RAM	GAME SIZE	PRICE
DISK	40K	8 worlds/1 black hole	$19.95
CASSETTE	32K	8 worlds/1 black hole	$14.95
DISK	32K	6 worlds/1 black hole	$19.95
CASSETTE	24K	6 worlds/1 black hole	$19.95
CASSETTE	24K	5 worlds/1 black hole	$ 9.95

Other offerings from Eastern Slope Computing:

•**Musical arithmetic practice** (grades 1-4)	$14.95
•**Letter Race** (grades 1-6)	$14.95
•**Graphics tutorial** (intro graphics aid)	$14.95
•**Fraction 500** (fraction estimation road race; grades 5-7)	$14.95
•**Demo disk** (includes complete version of all of the above - including Gates of Adventure)	$59.95

Eastern Slope Computing
545 South 43rd Street • Boulder, Colorado 80303

DEALER INQUIRIES WELCOME
Please add: $1.50 for shipping/handling • $1.00 for C.O.D.
Colo. residents add 5 1/2% sales tax ***Write for FREE flyer***

THE OSI® GAZETTE

OSI C1P Control Functions

C. A. Stewart
Adrian, MI

In a previous issue of **COMPUTE!** a basic poke version of my control function was published. Since that time I have discovered a method to implement a RUN command with a single key stroke much like the PET run key.

The main routine resides in page 2 in this revision (I used page 0 in my previous version but the added functions required relocation) and the one key screen clear resides in page 0. Refer to figure #1 (Flow chart) for discussion of the program functions.

In normal operations locations #536 and #537 contain vectors set by system ROM to the input routine $FFBA in typical 6502 hi/lo order i.e. $BA in location #536 and $FF in location #537. (Note for new computerist the symbol $ in machine language signifies HEX number, not string and the symbol # signifies a decimal number). By changing the vectors in these locations we force the system into our routine first and then return control to the ROM, to implement in this example we poke #536 with #128 and #537 with #002 (POKE536,128:POKE537,2) in one command line! It should be noted that a break warm start will require this poke command line since a warm start re-initializes these vectors.

A useful basic program for HEX to DEC and DEC to HEX is included in listing #3 for readers without tables or a TI HEX calculator.

When the routine starts we go to the input subroutine $0280 which jumps to $FFBA (input a character) and compare to the following.

Control L	**Load command**
Control S	**Save command**
Control A	**Run Command**
Escape Key	**List command**
Rubout Key	**Screen Clear**

If any of the comparisons are true then the appropriate subroutine in ROM is called, otherwise normal program operation continues. I chose Control A for the Run function for two reasons. First the logical choice, Control r, is utilized for a remove in the cursor control package I have in ROM and because of its location next to the control key. The command keys can be changed to whatever the user requires by replacing the compare data with the appropriate key numbers. Control A = $01 and follows thru with control Z = #26. (see graphics manual.)

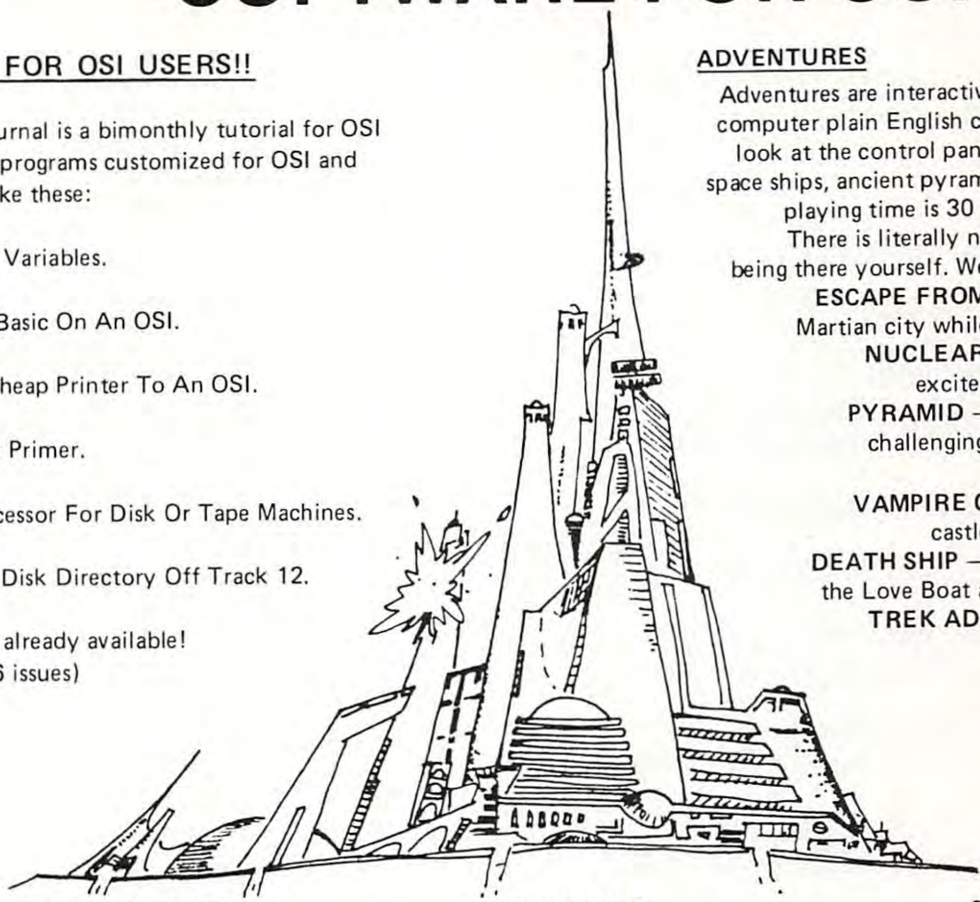
Listing #1 is the machine language routine. Listing #2 is the BASIC poke program. The machine language screen clear is callable in BASIC via the USR function. To use load and run, code erases itself, leaving the machine code in page 0 and 2, and doesn't require any normal usable memory.

```
D8              PHA
D9              LDA #$20;LOAD SPACE CHR
DB              LDX #$00;LOAD ACCX W/0
DD   HERE       STA $D000,X;STORE SPACE CH
R ON SCREEN
E0              STA $D100,X
E3              STA $D200,X
E6              STA $D300,X

E9              INX        ;INC X
EA              BNE HERE   ;BRANCH TO $DD I
F NOT EQUAL TO 0
EC              PLA
ED              RTS
```

```
280         JSR $FFBA;JUMP TO INPUT SUBRO
UTINE $FEED ON C4P
283         CMP #$0C ;COMPARE TO CONT L
285         BNE CONT S;BRANCH TO CONTROL
S
287         JSR $FF8B ;EXCUTE LOAD COMMAN
D
28A CONT S CMP #$13  ;COMPARE TO CONTROL
 S
28C         BNE RUB   ;
28E         JSR $FF96 ;EXCUTE SAVE COMMAN
D
291 RUB     CMP #$7F  ;COMPARE TO RUBOUT
293         BNE ESC   ;
295         JMP $D8   ;EXCUTE SCREEN CLEA
R
298 ESC     CMP #$1B  ;COMPARE W/ESCAPE
29A         BNE RUN   ;
29C         JMP $A4B5 ;EXCUTE LIST
29F RUN     CMP #$01  ;COMPARE W/CONT A
```

OSI # SOFTWARE FOR OSI **OSI**



A JOURNAL FOR OSI USERS!!

The Aardvark Journal is a bimonthly tutorial for OSI users. It features programs customized for OSI and has run articles like these:

1) Using String Variables.

2) High Speed Basic On An OSI.

3) Hooking a Cheap Printer To An OSI.

4) An OSI Disk Primer.

5) A Word Processor For Disk Or Tape Machines.

6) Moving The Disk Directory Off Track 12.

Four back issues already available!
$9.00 per year (6 issues)

ADVENTURES

Adventures are interactive fantasies where you give the computer plain English commands (i.e. take the sword, look at the control panel.) as you explore alien cities, space ships, ancient pyramids and sunken subs. Average playing time is 30 to 40 hours in several sessions. There is literally nothing else like them – except being there yourself. We have six adventures available.

ESCAPE FROM MARS – Explore an ancient Martian city while you prepare for your escape.

NUCLEAR SUBMARINE – Fast moving excitement at the bottom of the sea.

PYRAMID – Our most advanced and most challenging adventure. Takes place in our own special ancient pyramid.

VAMPIRE CASTLE – A day in old Drac's castle. But it's getting dark outside.

DEATH SHIP – It's a cruise ship – but it ain't the Love Boat and survival is far from certain.

TREK ADVENTURE – Takes place on a familiar starship. Almost as good as being there.

$14.95 each

NEW SUPPORT ROMS FOR BASIC IN ROM MACHINES

C1S – for the C1P only, this ROM adds full screen edit functions (insert, delete, change characters in a basic line.), Software selectable scroll windows, two instant screen clears (scroll window only and full screen.), software choice of OSI or standard keyboard format, Bell support, 600 Baud cassette support, and a few other features. It plugs in in place of the OSI ROM. NOTE: this ROM also supports video conversions for 24, 32, 48, or 64 characters per line. All that and it sells for a measly $39.95.

C1E/C2E for C1/C2/C4/C8 Basic in ROM machines.
This ROM adds full screen editing, software selectable scroll windows, keyboard correction (software selectable), and contains an extended machine code monitor. It has breakpoint utilities, machine code load and save, block memory move and hex dump utilities. A must for the machine code programmer replaces OSI support ROM. Specify system $59.95

DISK UTILITIES

SUPER COPY – Single Disk Copier
This copy program makes multiple copies, copies track zero, and copies all the tracks that your memory can hold at one time – up to 12 tracks at a pass. It's almost as fast as dual disk copying. – $15.95

MAXIPROSS (WORD PROCESSOR) – 65D polled keyboard only - has global and line edit, right and left margin justification, imbedded margin commands, choice of single, double or triple spacing, file access capabilities and all the features of a major word processor – and it's only $39.95.

P.C. BOARDS

MEMORY BOARDS!! – for the C1P. – and they contain parallel ports!
Aardvarks new memory board supports 8K of 2114's and has provision for a PIA to give a parallel ports! It sells as a bare board for $29.95. When assembled, the board plugs into the expansion connector on the 600 board. Available now!

PROM BURNER FOR THE C1P – Burns single supply 2716's. Bare board – $24.95.

MOTHER BOARD – Expand your expansion connector from one to five connectors or use it to adapt our C1P boards to your C4/8P. - $14.95.

ARCADE AND VIDEO GAMES

ALIEN INVADERS with machine code moves – for fast action. This is our best invaders yet. The disk version is so fast that we had to add selectable speeds to make it playable.
Tape - $10.95 – Disk - $12.95

TIME TREK (8K) – real time Startrek action. See your torpedoes move across the screen! Real graphics – no more scrolling displays. $9.95

STARFIGHTER – a real time space war where you face cruisers, battleships and fighters using a variety of weapons. Your screen contains working instrumentation and a real time display of the alien ships. $6.95 in black and white - $7.95 in color and sound.

MINOS – A game with amazing 3D graphics. You see a maze from the top, the screen blanks, and then you are in the maze at ground level, finding your way through on foot. Realistic enough to cause claustrophobia. – $12.95

SCREEN EDITORS

These programs all allow the editing of basic lines. All assume that you are using the standard OSI video display and polled keyboard.

C1P CURSOR CONTROL – A program that uses no RAM normally available to the system. (We hid it in unused space on page 2). It provides real backspace, insert, delete and replace functions and an optional instant screen clear.
$11.95

C2/4 CURSOR. This one uses 366 BYTES of RAM to provide a full screen editor. Edit and change lines on any part of the screen. (Basic in ROM systems only.)

FOR DISK SYSTEMS – (65D, polled keyboard and standard video only.)

SUPERDISK. Contains a basic text editor with functions similar to the above programs and also contains a renumberer, variable table maker, search and new BEXEC* programs. The BEXEC* provides a directory, create, delete, and change utilities on one track and is worth having by itself. – $24.95 on 5" disk - $26.95 on 8"

AARDVARK IS NOW AN OSI DEALER!

Now you can buy from people who can support your machine.

–THIS MONTH'S SPECIALS–

Superboard II	$279
C1P Model II	429
C4P	749

. . . and we'll include a free Text Editor Tape with each machine!

Video Modification Plans and P.C. Boards for C1P as low as $4.95

This is only a partial listing of what we have to offer. We now offer over 100 programs, data sheets, ROMS, and boards for OSI systems. Our $1.00 catalog lists it all and contains free program listings and programming hints to boot.

OSI **Aardvark Technical Services • 1690 Bolton • Walled Lake, MI 48088**
(313) 669-3110 **OSI**

```
2A1        BNE END   ;
2A3        JSR $A477 ;INITALIZE ROUTINE
2A6        JSR $A5C2 ;RUN ROUTINE
2A9 END    RTS       ;
```

Listing 1

```
 10 REM MACHINE LANG SUBROUTINE FOR OSI
C1P/C4P
 20 REM CHARLES A. STEWART
 30 REM 3033 MARVIN DR.
 40 REM ADRIAN, MICH 49221
 60 FORX=640TO681:READA:POKEX,A:NEXT
 65 REM FOR C4P LINE 70 CHANGE 186,255 T
O 237,254
 70 DATA32,186,255,201,12,208,3,32,139,2
55,201
 80 DATA19,208,3,32,150,255,201,127,208,
3,76
 90 DATA216,0,201,27,208,3,76,181,164
 100 DATA201,1,208,6,32,119,164,32,194,1
65,96
 110 FORX=216TO237:READA:POKEX,A:NEXT
 120 DATA72,169,32,162,0,157,0,208,157,0
,209,157,0,210
 130 DATA157,0,211,232,208,241,104,96
 150 POKE11,216:POKE12,0:POKE536,128:POK
E537,2
 160 PRINT"*CONTROL VERSION #1":PRINT"*B
Y CHARLES A. STEWART"
 165 PRINT:PRINT
 170 PRINT"ESC LISTS":PRINT"RUBOUT GIVES
 SCREEN CLEAR
 180 PRINT"CONTROL S =SAVE":PRINT"CONTRO
L L = LOAD
 185 PRINT"CONTROL A RUNS PROGRAM
 200 NEW
```

Listing 2

```
 10 REM CHARLES A. STEWART
 20 REM 3033 MARVIN DR.
 30 REM ADRIAN MI 49221
 40 REM 517-265-4798
 50 REM NOVEMBER 22, 1980
 60 REM DEC TO HEX AND HEX TO DEC CONVER
SON PROGRAM
 100 DIMA$(16),S$(16):FORX=1TO16:READA$(
X):READS$(X):NEXT
 110 DATA0000,0,0001,1,0010,2,0011,3,010
0,4,0101,5,0110,6
 120 DATA0111,7,1000,8,1001,9,1010,A,101
1,B,1100,C,1101,D,1110,E
 130 DATA1111,F
 135 S$="0123456789ABCDEF"
 140 POKE11,0:POKE12,253:X=0:Y=0:W=0:Q=0
:I=0:E$="0"
 150 FORX=0TO40:PRINT:NEXT:PRINT"A> DECI
MAL TO HEX":PRINT
 160 PRINT"B> HEX TO DECIMAL":PRINT:PRIN
```

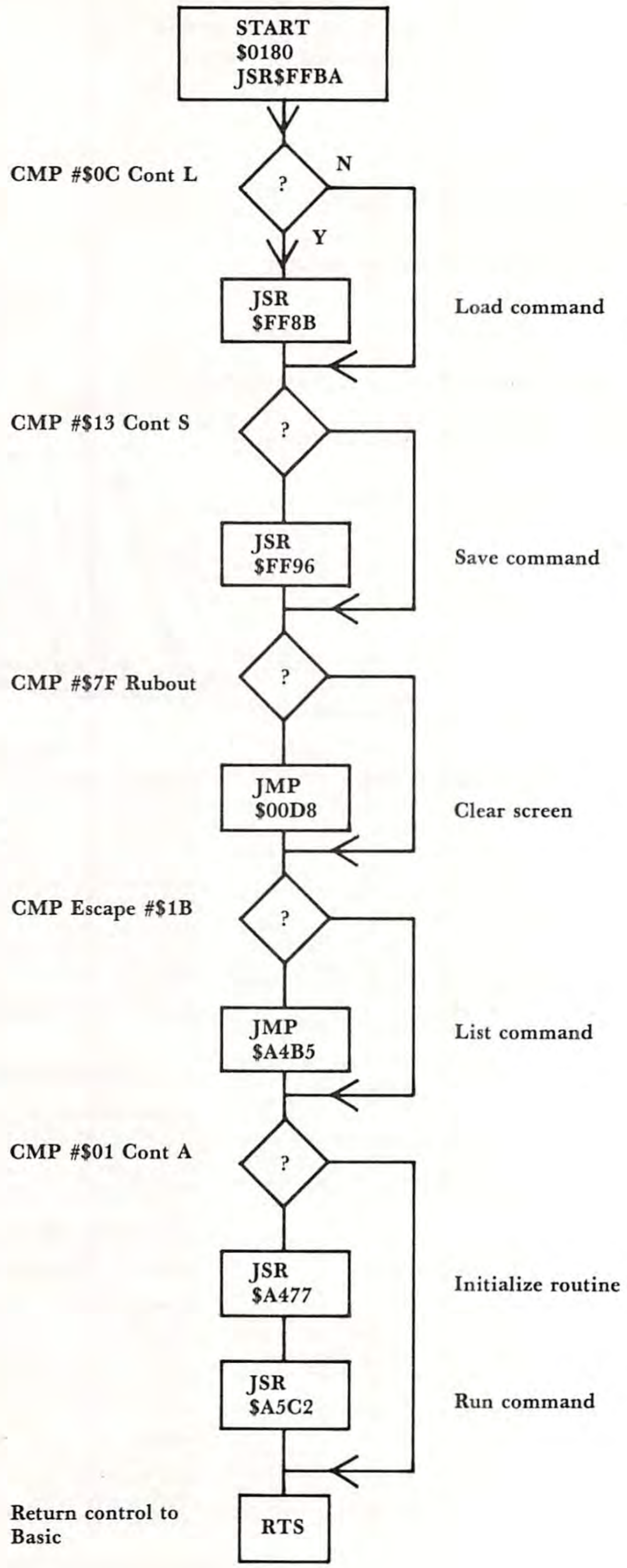


Figure #1

```
T"YOUR SELECTION":X=USR(X)
 170 IFPEEK(531)=65THENPRINT"DEC TO HEX
CONVERSION":GOTO3010
 180 IFPEEK(531)=66THENPRINT"HEX TO DEC
CONVERSION":GOTO2010
 190 GOTO150
 2010 PRINT:INPUT"HEX NUMBER";I$:IFLEN(I
$)>4THEN2010
 2020 IFLEN(I$)<>4THENI$=E$+I$:GOTO2020
 2040 FORX=1TO4:FORY=1TO16
 2050 IFMID$(I$,X,1)=MID$(S$,Y,1)THENB$(
X)=A$(Y)
 2060 NEXTY:NEXTX
 2070 B1$=B$(1)+B$(2)+B$(3)+B$(4)
 2080 PRINT:PRINTI$" IN BINARY=":PRINTB1
$
 2100 X=1:W=0:Q=LEN(B1$):I=0
 2120 Y$=MID$(B1$,Q,1):Y=VAL(Y$):I=Y*X:W
=W+I:X=X*2
 2130 Q=Q-1:IFQ<>0GOTO2120
 2140 PRINT:PRINTI$" IN DECIMAL=";W
 2150 PRINT:PRINT"TYPE ANY KEY TO CONTIN
UE":X=USR(X):GOTO2010
 3010 PRINT:INPUT"DECIMAL NUMBER";I$:I=V
AL(I$):Y$=" ":Y=65536
 3012 Y=Y/2
 3015 IFI>65535THENPRINT:PRINT"TOO LARGE
":GOTO2150
 3030 X=INT(I/Y):IFX=0THENY$=Y$+"0":GOTO
3050
 3040 Y$=Y$+"1":I=I-Y
 3050 Y=Y/2:IFINT(Y)=0THEN3200
 3060 GOTO3030
 3200 PRINT:PRINTI$"IN BINARY=":PRINTY$
 3210 X=2:Y=4
 3215 RE$=" "
 3220 A$=MID$(Y$,X,Y):FORW=1TO16:IFA$=A$
(W)THENRE$=RE$+S$(W):GOTO3240
 3230 NEXTW
 3240 X=X+4:IFX>14THENGOTO3260
 3250 GOTO3220
 3260 PRINT:PRINTI$" IN HEX=";RE$
 3265 PRINT"TYPE ANY KEY TO CONTINUE":X=
USR(X):GOTO3010
 OK
```

Listing 3 ©

Double The Cassette Baud Rate Of Your OSI Superboard II/1P

Mr. James L. Mason
Jacobus, Pennsylvania

I was thrilled when I received my Superboard II. It was my first micro, but being experienced in BASIC Programming (using a phone-linked ASR 33 with GE time-sharing) I had several programs which I was anxious to try. The Superboard performed perfectly. It did everything OSI said it would, however, 2 disadvantages of Superboard soon made themselves apparent. The 25 x 25 character video format was not the easiest to read. Secondly, the baud rate at which programs are saved and loaded from cassette seemed painfully slow. Having a good working background in digital electronics, I thought it might be possible to improve upon these two features. Upon close examination I found the video hardware was too intimate with the software in ROM. Fortunately, modifying the cassette port circuitry was a piece of cake and I was able to cut load and save time by half.

The cassette port utilizes a 6850 programable **A**synchronous **C**ommunications **I**nterface **A**dapter (see figure 1). When using this chip, the communications rate is determined by two things, the frequency of the clock which is applied to the TXCLK and RXCLK pins of the ACIA and the control word which is written into the ACIA's control register. I hypothesied that by doubling the clock frequency I could double the baud rate.

ON the Superboard, a crystal oscillator generates the base timing signal by which the entire board is controlled. This signal drives a synchronous divider chain (see figure 2). The timing signal destined for the ACIA comes off the ÷ 32 tap of this chain. The signal is finally divided by a ÷ 24 circuit composed of U57, U58, and U63. The resultant frequency of 5120 Hz is applied to the TXCLK input of the ACIA. The ACIA must be programmed to utilize a clock frequency either 64, 16, or 1 times the baud rate. The 320 baud rate is realized by programming the ACIA for a 16x clock rate.

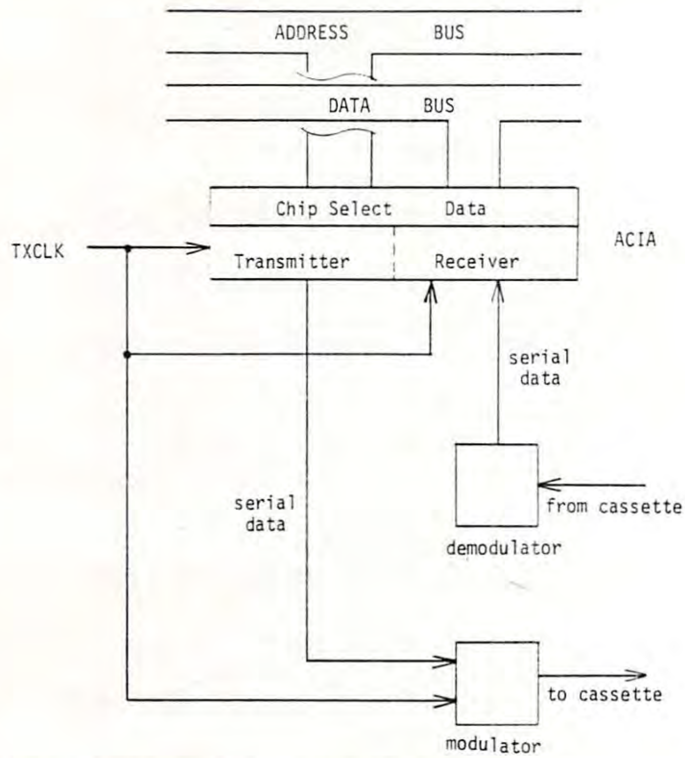


Figure 1. Cassette Port Block Diagram

To obtain a clock rate double of that which is used, I chose to sever the connection between U57 pin 2 and U59 pin 14 (see figure 3), then connect U57 pin 2 to U30 pin 11. I used a switch to maintain compatibility with my old 320 baud tapes.

As far as the ACIA was concerned, the modification was done. However, there is one more block between the ACIA and the cassette machine, namely the Modulator/Demodulator.

The modulator encodes the data in the form of tones. These tones are derived from the TXCLK (see Figure 4A). Since our new TXCLK is twice as fast, our tones will now be 2 times their original frequency. This poses no problem as far as modulation is concerned. It does, however, make a difference on the return trip. U69 determines what will be demodulated as a high or low tone (see figure 4B). A tone coming in will trigger the 74123 one-shot by it's rising edge (see figure 5). R57 is adjusted so that U69 will remain triggered until after the falling edge of the high frequency tone but not until the falling edge of a low frequency tone. The falling edge will clock the D flop U63 and propagate the state of U69. Because we now have shorter pulse widths, R57 must be adjusted to allow U69 to time-out during our new low tones. This was very simple to do. I simply saved a program using the new faster baud rate and attempted to load it back. While the program was trying to load, I adjusted R57 while watching the video monitor. I knew I had R57 adjusted properly when the program began appearing on the display, line by line. I experimented with R57 to find the points where data started to be garbled. The margin was surprisingly wide. Luckily, no software patch had to be made anywhere.

If you use a switch in your mod, remember you will have to readjust R57 each time you change baud rates. I see no reason why a DPDT switch couldn't be used to switch in a different resistor value for R57 along with switching the clock rate.

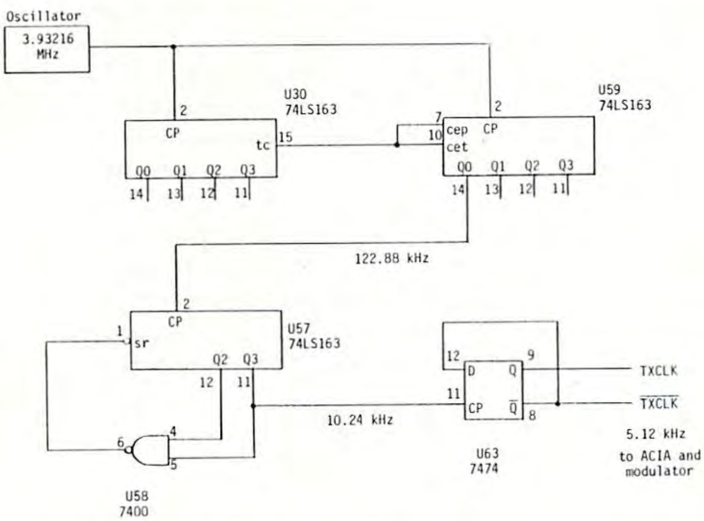


Figure 2. TXCLK Generation

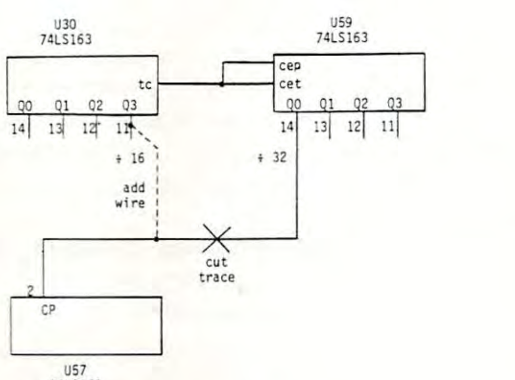


Figure 3. Installing The Modification

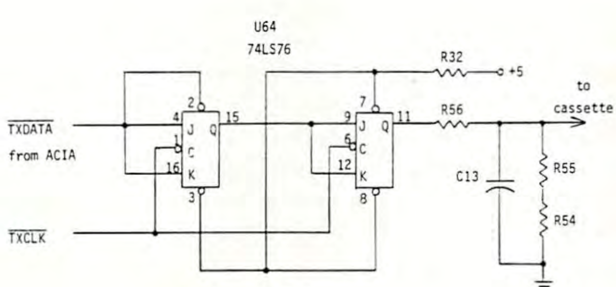


Figure 4A. Modulator

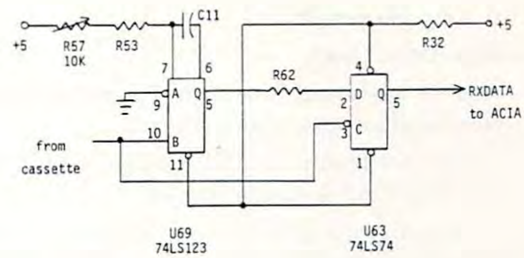


Figure 4B. Demodulator

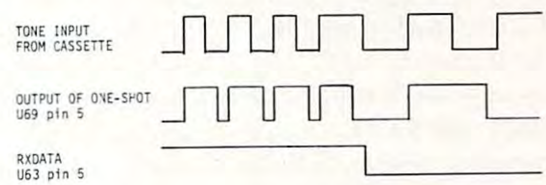


Figure 5. Demodulator Timing ©

Review

HEXDOS 2.3: A Disk Operating System For The OSI C1P or Superboard II

Ronald C. Whitaker
Salt Lake City, Utah

The day I received my OSI disk drive and 610 Expander board, I hooked them to my C1P and my homebuilt power supply, turned them on, and pushed "D" to boot up the disk. OSI's Pico-Dos came with the drive and expander board and booted up OK but would only allow me to load eight programs of up to 8K each. This was faster than cassette to be sure, but definitely lacking the features I wanted in a disk operating system. I lacked funds for OSI's OS65D and the additional 4K of memory it required to run on my system. The future looked dismal, indeed!

The day was saved by a single stroke of good fortune. Several months earlier a local dealer had loaned me a catalog from "The 6502 Program Exchange". While oriented mostly toward Apple and single board systems, they did have a few OSI compatible programs listed. One of these was HEXDOS 2.3 for the C1P and Superboard II by Steve Hendrix. The features promised by the catalog sounded too good to be true! These included:

1. An operating system and directory which occupied only the first two tracks on the disk

2. A tone generator
3. Real-time Clock
4. Special keyboard functions for instant screen clear and program line editing.
5. Utility programs to CREATE, DELETE, and FORMAT program space on the disk

I sent for the program and received it in about 10 days. My first attempts to use it were frustrating. It would function erratically or not at all. But in a quick exchange of correspondance with the author, the problem was resolved, my copy of the program was updated and has functioned flawlessly ever since. It is an exceptionally compact system due to its full use of ROM BASIC and the ROM Monitor software built into the C1P. The following is a brief discussion of the features I have found most useful.

1. **LOAD and SAVE** commands are the heart of any operating system. HEXDOS uses LOAD (filename) and SAVE (filename) to load and save programs by filename. Other commands open and close named data files, each consisting of a 2K block of memory which fills one disk track. Opening a data file reserves a 2K block of RAM for that file. Closing a file loads the contents of that 2K block to one track of the disk. Each file is designated as an input file or as an output file and up to 11 of each are allowed simultaneously, provided there is sufficient RAM memory to support them.

The most versatile and, potentially, the most useful LOAD and SAVE commands allow the user to load 2K bytes from a specified track to any location in RAM memory. The corresponding save command loads any 2K block of memory to be saved to a specified track number. Because of the directness of these commands, care must be used to avoid accidental over-writing of existing data or program memory. Using these direct LOAD and SAVE commands, I have written programs using record and file lengths of my own choosing.

2. **INPUT and PRINT** commands control I/O to or from disk data files, video screen, keyboard, 6850 ACIA, and ports reserved for printer and modem.

3. **SPECIAL KEY FUNCTIONS** make use of unused keys to provide instant screen clear, suspension of output until key is released, break in a BASIC program, and non-destructive forward and backward movement of the cursor for simple line editing. These functions are so useful that they are sorely missed on those rare occasions when I'm writing a program without HEXDOS.

4. **SINGLE STEP and TRACE** functions are added using simple POKE statements, and are very useful for debugging programs.

5. **The USR** function is used to provide several special functions:
   a. Control of a tone generator with 256 different tones
   b. Input of a character from one of 256 possible input devices
   c. Return contents of the real-time clock
   d. Direct jump to a machine code routine located anywhere in memory
   e. Jump to ROM monitor
   f. Jump to the last machine code routine loaded from the disk

Use of the tone generator and the real-time clock requires very simple hardware modifications which are well detailed in the HEXDOS manual and require only a few minutes to perform. The real-time clock requires a single jumper between pads provided on the 610 board. Since I have my C1P interfaced to a General Instrument's Programmable Sound Generator, I have not tried the tone generator, but it requires a single jumper and an audio amplifier. Additional hardware modifications include repositioning the break key to a less vulnerable location, automatic power-on reset (D/C/W/M appear when the computer is turned on), and disk drive motor control, which turns off the drive motor when it is not being used. None of these modifications are necessary for HEXDOS operation but all of them are useful additions to the computer's capabilities.

6. **UTILITY PROGRAMS:**

FORMAT will completely erase a disk and then format it for HEXDOS by loading the operating system onto track 0 and reserving track 1 for the directory.

CREATE names and reserves any number of tracks for use as data files or for storage of machine language programs, which require filenames beginning with $. These can then be loaded directly into any specified location in user RAM.

DELETE erases any program listed in the directory, deallocates the space on the disk reserved for it, and repacks the remaining files.

DISASSEMBLER lists machine code programs using standard 6502 mnemonics and identifies the addressing mode of the instruction.

HEXDOS 2.3 is an extremely compact, easy-to-use, and versatile operating system for the OSI C1P or Superboard II. Because it is so compact it can be placed at the beginning of every disk. Documentation is clear and complete, the best I have seen from any source. Because I knew nothing about data files or their use, I wrote to the author of HEXDOS, Steve Hendrix, for information. Besides a personal reply, I received two new pages of documentation explaining use of disk files. Using HEXDOS 2.3 has been a pleasant and rewarding experience for me and I strongly recommend it as a versatile and inexpensive alternative to other operating systems.

HEXDOS 2.3 is available for $25 from:
The 6502 Program Exchange
2920 West Moana Lane
Reno, NV 89509 ©

THE PET® GAZETTE

Partition and Load

R. D. Young
Ottawa, Ontario

And another mystery is solved. This one began where I left off in "Relocate" (**COMPUTE!**, Issue 9, Feb. 1981, p. 103). I think that I have now consolidated Harvey B. Herman's memory partitioning and Charles Brannon's "Quadra-PET" into one routine that makes use of "Relocate" as well. I must acknowledge with gratitude Jim Butterfield's most helpful hints, particularly the various memory maps published over the months, and Harvey B. Herman's memory partitioning challenge in the first place, coupled with his version of "Relocate" that helped to translate for the Upgrade ROM Version.

Note that I have provided two listings to accomodate original and upgrade ROMs. I have not attempted to provide for the 80-column machine. The BASIC program performs two functions: it places a machine language program into the second cassette buffer, and it initializes the partitions and the pointer storage locations (1010 to 1017 decimal). The BASIC program ends with a NEW statement, leaving the PET in the lowest partition which is equivalent to the old standard 8K PET (FRE(0) = 7164).

I chose to use a BASIC program for this loading and initialization after trying several other alternatives. Its main advantage is that the machine language program could be made small enough to fit in the second cassette buffer with the advantages that second cassette buffer use offers. Some problems with loading and initialization were also alleviated. However, because of space restrictions, error checking of user input is not performed. In other words, if you exceed the input limits *** **CRASH** ***.

The machine language program provides control over four (4) memory partitions. It will function with 16K, 24K, or 32K PETs, providing, of course, that the proper inputs are used. That's twice now that I've mentioned inputs. Here is how the routine is used:

1. The BASIC program has left you in Area 1 (low).
2. Use SYS 826 to call the routine. A flashing cursor will appear below the SYS826 on the screen. This is asking you for your input (number 1 to 7) according to the following options:
   1 - go to Area 1 (low 7K)
   2 - go to Area 2 (next 8K)
   3 - go to Area 3 (next 8K)
   4 - go to Area 4 (high 8K)
   5 - load Area 2
   6 - load Area 3
   7 - load Area 4
3. To load Area 1, go to Area 1 and LOAD in the normal way.
4. Key in and SAVE programs from Area 1.
5. Clear all partitions by POKE 135,maximum:POKE 123,4:NEW — or upgrade ROM equivalents.

There are a few precautions to be observed. Load/ Verify and device flags are not set by this routine. To load from Tape #1, for example, you must set these flags manually (see "Hooray for SYS", **COMPUTE!**, Issue 8, Jan. 1981, p. 96), or you must have previously loaded a program from Tape #1. This routine assumes that BASIC pointers in locations 122 and 134 (original ROM) are never changed. And lastly, going from one partition to another will result in the loss of your program *if* the program is less than 256 bytes long. Also remember to observe the input limits for the memory size in your machine.

Should you want or need to find out where the partitions are, the pointers to the end of BASIC in each partition restored in locations 1010 to 1017 decimal as follows (original ROM):

1010	124	Area 1
1011	125	Area 1
1012	124	Area 2
1013	125	Area 2
1014	124	Area 3
1015	125	Area 3
1016	124	Area 4
1017	125	Area 4

In fairness to all, I should mention that my PET is equipped with the original ROM only, and has 16K of memory. The routine functions with my PET, but there could still be some bugs remaining for obvious reasons.

I am looking forward to seeing any comments on the routine in **COMPUTE!**. For now, I'm fresh out of mysteries. ©

Skyles Electric Works

BASIC Programmer's, Toolkit™, Disk-O-Pro™, Command-O™

For PET™ Owners Who Want More Fun And Fewer Errors with Their Programs

Here are thirty-five commands you'll need, all on dual chips you can install in two minutes without tools, **on any PET or PET system.** 2KB or 4KB of ROM firmware on each chip with a collection of machine language programs available from the time you turn on your PET to the time you shut it off. No tape to load or to interfere with any running programs.

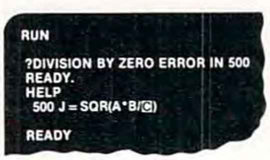
For PET/CBM 2001-8, -8N, -16N/B, -32N/B, 3016 and 3032

BASIC Programmers Toolkit™ commands

AUTO^{ed} DELETE^{ed} RENUMBER^{ed} HELP^{ed} TRACE^{ed} STEP^{ed} OFF^{ed} APPEND^{ed} DUMP^{ed} FIND^{ed}

BASIC Programmers Disk-O-Pro™

CONCAT^{B80} DOPEN^{B80} DCLOSE^{B80} RECORD^{B80} HEADER^{B80} COLLECT^{B80} BACKUP^{B80} COPY^{B80} APPEND^{B80} DSAVE^{B80} DLOAD^{B80} CATALOG^{B80} RENAME^{B80} SCRATCH^{B80} DIRECTORY^{B80} INITIALIZE^{BS} MERGE^{BS} EXECUTE^{BS} SCROLL^{ed} OUT^{ed} SET^{ed} KILL^{ed} EAT^{ed} PRINT USING^{BS} SEND^{BS} BEEP^{BS}



NOTES:

ed — a program editing and debugging command
B80 — a BASIC command also available on Commodore CBM™ 8016 and 8032 computers.
BS — a Skyles Electric Works added value BASIC command.
BASIC Programmers Toolkit™ is a trademark of Palo Alto IC's.
BASIC Programmers Disk-O-Pro™, Command-O™ are trademarks of Skyles Electric Works.
PET™, CBM™ are trademarks of Commodore Business Machines.

AVAILABLE: USA/CANADA: Please contact your local dealer
England: Calco Software Lakeside House, Kingston Hill, Surrey KT2 7QT
GERMANY: Unternehmensberatung, Axel Brocker Lennebergestr 4, 6500 Mainz
Japan: Systems Formulate, 1-8-17 Yaesu Shinmaki-cho Bldg. 11F Chuo-ku, Tokyo JAPAN 103

Phone or write for information. We'll be delighted to answer any questions and to send you the complete information package.

Skyles Electric Works
231 E South Whisman Road
Mountain View, CA 94041
(415) 965-1735

```
1Ø REM *** PARTITION & LOAD LOADER ***
2Ø REM *** AND INITIALIZATION ***
25 REM *** ORIGINAL ROM VERSION ***
3Ø REM *** BY R.D. YOUNG ***
4Ø REM
5Ø REM
1ØØ REM *** LOAD 2ND CASSETTE BUFFER ***
11Ø PRINT"♥"
12Ø FOR I=826 TO 990: READ A: POKE I,A: NEXT
13Ø REM
14Ø REM *** INITIALIZATION ***
15Ø POKE 1Ø12,4: POKE 1Ø14,4: POKE 1Ø16,4
16Ø POKE 1Ø13,32: POKE 1Ø15,64: POKE 1Ø17,96
17Ø POKE 135,32: POKE 8192,Ø: POKE 16384,Ø: POKE 24576,Ø
18Ø NEW
19Ø REM
1ØØØ DATA 32,223,241,56,233,48,17Ø,141,241,3,189,215,3,133,123,188
1Ø1Ø DATA 2Ø1,3,32,176,3,165,124,157,241,3,165,125,157,242,3,152
1Ø2Ø DATA 17Ø,189,241,3,133,124,189,242,3,133,125,174,241,3,189,2Ø8
1Ø3Ø DATA 3,133,135,224,4,16,12,165,125,197,123,24Ø,3,76,1Ø6,197
1Ø4Ø DATA 76,83,197,169,147,32,234,227,32,174,245,173,126,2,56,237
1Ø5Ø DATA 124,2,141,241,3,165,123,141,124,2,24,1Ø9,241,3,141,126
1Ø6Ø DATA 2,169,55,141,24,129,141,25,129,169,1,141,13,2,169,13
1Ø7Ø DATA 141,15,2,76,195,243,165,135,2Ø1,32,2Ø8,3,162,1,96,2Ø1
1Ø8Ø DATA 64,2Ø8,3,162,3,96,2Ø1,96,2Ø8,3,162,5,96,162,7,96
1Ø9Ø DATA 1,3,5,7,3,5,7,32,64,96,128,64,96,128,4,32
11ØØ DATA 64,96,32,64,96
```

```
1Ø REM *** PARTITION & LOAD LOADER ***
2Ø REM *** AND INITIALIZATION ***
25 REM *** UPGRADE ROM VERSION ***
3Ø REM *** BY R.D. YOUNG ***
4Ø REM
5Ø REM
1ØØ REM *** LOAD 2ND CASSETTE BUFFER ***
11Ø PRINT"♥"
12Ø FOR I=826 TO 989: READ A: POKE I,A: NEXT
13Ø REM
14Ø REM *** INITIALIZATION ***
15Ø POKE 1Ø12,4: POKE 1Ø14,4: POKE 1Ø16,4
16Ø POKE 1Ø13,32: POKE 1Ø15,64: POKE 1Ø17,96
17Ø POKE 53,32: POKE 8192,Ø: POKE 16384,Ø: POKE 24576,Ø
18Ø NEW
19Ø REM
1ØØØ DATA 32,2Ø7,255,56,233,48,17Ø,141,241,3,189,214,3,133,41,188
1Ø1Ø DATA 2ØØ,3,32,175,3,165,42,157,241,3,165,43,157,242,3,152
1Ø2Ø DATA 17Ø,189,241,3,133,42,189,242,3,133,43,174,241,3,189,2Ø7
1Ø3Ø DATA 3,133,53,224,4,16,12,165,43,197,41,24Ø,3,76,117,197
1Ø4Ø DATA 76,93,197,169,147,32,216,227,32,166,245,173,126,2,56,237
1Ø5Ø DATA 124,2,141,241,3,165,41,141,124,2,24,1Ø9,241,3,141,126
1Ø6Ø DATA 2,169,55,141,24,129,141,25,129,169,1,133,158,169,13,141
1Ø7Ø DATA 111,2,76,185,243,165,53,2Ø1,32,2Ø8,3,162,1,96,2Ø1,64
1Ø8Ø DATA 2Ø8,3,162,3,96,2Ø1,96,2Ø8,3,162,5,96,162,7,96,1
1Ø9Ø DATA 3,5,7,3,5,7,32,64,96,128,64,96,128,4,32,64
11ØØ DATA 96,32,64,96
```

©

Last Minute Correction From R. D. Young

With reference to my program, "Partition and Load", dated February 15, 1981, there is a correction to be made in the program listings. In the DATA statement at line 1030 of both listings, the '4' between '224' and '16' must be changed to a '5'.

There is also one other precaution to be observed, although it is a function of the imbedded "Relocate" routine. A program loaded with this routine should not have a line 77 as part of the program. Any line 77 will be deleted because of the dynamic RETURN feature used to reset line links after the LOAD.

Relative File Mechanics

Jim Butterfield, Toronto

I'll use the Basic 4.0 commands, since they are more convenient.

Creating a Relative File

To create a new Relative file, we may use Direct Commands if we wish. This is the easiest way to see how it all happens — try it Directly and later you can incorporate it into a program if you wish. Suppose we want a file of 25 items (initially) each of which may be up to 20 characters long.

DOPEN#1, "RFILE",L20 opens the file for relative writing. We've specified a length of 20 characters.

RECORD#1, 25 positions us to the last record desired, number 25, but this doesn't exist yet. The error light on the disk will turn on; we find a RECORD NOT PRESENT signalled. If we tried INPUT or GET at this time, we'd be in serious trouble. But we can still write, and this will create all records up to and including number 25.

PRINT#1 writes a Return character in record 25 of the file, and incidentally creates all records up to and including 25. You'll notice a delay of several seconds while this happens.

DCLOSE#1 wraps up the file in the usual way.

Writing to a relative file

If you have gone through the DOPEN/RECORD/PRINT/DCLOSE sequence above, we can use this file to write some data. Once again, let's use Direct statements to allow us to watch things happening.

DOPEN#1, "RFILE" opens our file for reading or writing. Note that we don't need to specify the file type (REL) or whether we wish to read or write (we might do either). And we must not specify the length L — that's only for creating the file.

RECORD#1,10 positions us so that we can read or write record number 10. There's no error light this time, since record 10 is in place. Note that if our record number was a variable such as X, we'd need to put it in brackets, e.g., RECORD#1,(X). Now let's write a few records starting at item ten.

PRINT#1, "HELLO" writes six characters (HELLO plus the RETURN character) to record number 10; the disk automatically positions to the start of record 11. This positioning is not triggered by the Return character — it's done by recognizing the end of PET's transmission (technically speaking, the EOI line).

PRINT#1, "THERE"; uses a semicolon to supress Return so only five characters are written to record 11. End of transmission is correctly detected, however, and the disk positions to record 12.

PRINT#1, "A" + 7CHR$(13) + "B" writes a single record, even though there are two Returns. When we are reading, it will take two INPUT# statements to get the information, since INPUT stops on a Return character.

PRINT#1, "THIR" + "TEEN" writes THIRTEEN in the usual way. **PRINT#1, "FOUR"; "TEEN"** writes FOURTEEN.

PRINT#1, "MORE THAN TWENTY CHARACTERS" will generate a disk error, "Overflow in Record", which tells you that you're trying to fit too much into a record. The first twenty characters will be written to the record, the rest discarded.

DCLOSE#1 terminates our writing session.

Reading from a relative file

We must have a program to GET# or INPUT# our data; these commands can't be given as Direct statements.

```
100 DOPEN#1,"RFILE"
110 INPUT "RECORD NUMBER DESIRED";R
120 IF R = 0 GOTO 190
130 RECORD#1,(R)
140 IF DS>0 THEN PRINT DS$:GOTO 110
150 INPUT#1,R$
160 PRINT R$
170 IF ST = 0 GOTO 150
180 GOTO 110
190 DCLOSE#1
```

Use the above program to browse through the items we have just written. You'll find some interesting things. For example, what do you see in records you have never written (say, record 4)? What happens if you try for record number 200? Does it seem to matter if you have written a Return to the record or not?

A new role for ST, the Status Byte

About the ST test in line 170: you may recall that we created a record with more than one item. Record 12 had two items (called "fields") each followed by a Return character. An INPUT# statement will stop on the first Return and would not see the second field.

But ST works in a very useful way: it is set to 64 (end of file) at the end of every record. That means that if we might have more than one field in a record, we can check ST to see if we have read the last one.

One drawback of this mechanism is that you can't use ST to tell you if you are at the end of a relative file; it flags EOF after every record. How do you tell when you're at the end? Use the RECORD command; when you get an error 50, RECORD NOT PRESENT, you'll know you are past the end.

Enlarging a file

Easy. Use RECORD followed by PRINT# in the same manner as when you created the file. The new records will automatically be written up to the record number you have specified.

Conclusion

There are new rules to learn. But Relative files are easy to use, and add power and speed to your programs. ©

Odds and Ends

Charles A. McCarthy
St. Paul MN

Microsoft Basic for the PET, and probably for other 6502 machines, treats multiplication $z = x*y$ asymmetrically. The execution time depends upon the binary representation of the factor y, and not on x. This can be inferred from studying the coding of the multiplication routine, and is proved by experiment. On my PET, the execution time for one multiplication $z = x*y$ is about 3840 + 30 A + 230 B microseconds where A is the number of non-zero bits in y and B is the number of non-zero bytes in y.

As a practical consequence, if you know that one of your factors must have only one or two non-zero bytes, it should be placed as y. When $x = 3$ and $y = \pi$, the execution time is about the same as when $x = \pi$ and $y = \pi$, and is about 20% longer than when $x = \pi$ and $y = 3$. ©

PETTERM™

Terminal Program

Intelligent Communication

Uploads and Downloads

In Machine Language

Use your PET as an intelligent terminal. Access timeshare systems and networks such as the Source or Telenet. This is an interrupt driven system with buffered input/output. Unlike dumb terminals, the PET can do other tasks instead of having to wait for incoming data. This flexibility allows many advanced features.

These are complete assembled hardware and software packages. All include line editing/resend, auto-repeat, shift lock, output to IEEE printer and much more. Delivered on cassette or disk. Also, available for Commodore 8010 modem or Livermore Star 488 modem.

PETTERM I All features above .... **$ 80.00**
PETTERM II All features of I. plus local text editor with down-loading capability . . **90.00**
PETTERM III All features of II. plus 80 132 column scrolling window for viewing formatted outputs wider than 40 columns **100.00**

FORTH

A Programming Revolution

FORTH Interactive high level compiler and operating system 5–10 times faster than PET BASIC. High level block structured language. This is a true fig-FORTH implementation for 16 and 32K PETs.

fig-FORTH, with editor, assembler **$50.00**
with floating point and strings **$60.00**
with floating point, strings, macro-assembler . . **$70.00**
cassette and disk interface available.

DEALERS INQUIRE

F S S

software for small computers

1903 Rio Grande
Austin, Texas
78705

1-512-477-2207

P.O. Box 8403
Austin, Texas
78712

Source: TCE995

COPLOT

Anselm Wachtel
Pittsburgh, PA

This program, entitled "COPLOT" incorporates two previously published routines: 1. A plotting routine by J. S. Barney, Pet User Notes, July/August 1978, page 1; 2. An automatic selection of plotting limits routine by Wm. D. Johnston, Microcomputing, May 1980, page 26.

The former raises the normal resolution by a factor of 8 (singe-pixel resolution) in any one direction. I have chosen the vertical (Y) direction for the function values, and both for the positioning of the coordinate leaders.

The latter receives as input the numeric variable limits and the number of divisions which are desired on the axis. It then returns a new rounded-up interval for the limits, as well as the value of the interval in the same units. In the present program, it is used twice, once for each axis.

As in any program of this type, calculation of f(x) can lead to illegal quantities such as division by 0 or log(-x), etc. Since the function is entered (or edited) in line 120 as DEFNA(X) = . . . by the user, there is no way to anticipate which values will cause an error message in the course of scanning from the low to the high X-bounds, and the user has to anticipate this by applying elementary considerations. For instance, if DEFNA(X) = 1/(X*X – 5*X↑14) then a ?DIVISION BY 0 ERROR will occur at x = –2 if the X bounds are, say, –5 and 3, and at x = 7 if they are 3 and 10. If the function is log of the above, then the entire range from –2 to +7 is illegal. The function is evaluated at line 290, and an appropriate line 285 such as IF X*X – 5*X – 14 = 0 THEN NEXTI should take care of that.

Notwithstanding the high resolution which leads to attractive looking displays, the program is not intended to allow an exact evaluation of function values, e.g. replace graph paper ruled in mm or 1/20 inch divisions. Theoretically, the values are located at the *left* ends of the bars and this, moreover, can create the illusion of a displacement in cases where a symmetrical curve is known to peak at some known value, because the viewer (unintentionally) tends to look at the center of the bars. In order to avoid overprinting of labels, the number of divisions are limited to 8 in the abscissa and 10 on the ordinate. At least on the abscissa, anything denser would create confusion even in the case of integer-labels, because these are not in good registry with the high-resolution leaders, especially considering the leading space which the PET places in front of positive numbers, including 0.

All things considered, this program should prove to be a valuable aid to any one desiring a quick sketch of some function within any reasonable limits.

```
0 REM PLOT ROUTINE BASED ON PROGRAM BY ¬
     ¬J.S.BARNEY - SEE PET USER NOTES
10 REM  JULY/AUGUST, 1978, PAGE 1,
     ¬ AND AUTOMATIC SELECTION OF ¬
     ¬PLOTTING LIMITS
20 REM BY WM. D. JOHNSTON, SEE MICROCOMP
     ¬UTING MAY,1980, PAGE 26.
30 REM ADAPTED FOR SCALING OF COORDINATE
     ¬S AND FUNCTIONS BY A.WACHTEL,
40 REM PITTSBURGH, PA.  AUGUST,1980.
50 REM ENTER FUNCTION IN LINE 120. EDIT ¬
     ¬AT END AFTER HITTING 'F'.
60 REM INPUT NEW X-INTERVAL AT END ¬
     ¬AFTER HITTING ANY OTHER KEY.
70 D$="↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓":REM 20 ¬
     ¬CURSOR DOWN.
80 X$="→→→→→→→→→→→→→→→→→→→→→→→→→→→→→→→→"
     ¬:REM 32 CURSOR RIGHT.
90 Y$="↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑":REM 20 ¬
     ¬CURSOR UP.
100 V$="$RF@CDE#":REM SHIFT"$RF@CDE#"
110 U$="%TGB]HY'":REM SHIFT "%TGB]HY'"
120 DEFFNA(X)=SIN(X)
130 DIM X(32),Y(32),S(12)
140 INPUT"ħLOWER, UPPER X-BOUNDS";A1,B1
150 AX=A1:BX=B1
160 PRINT"↓# OF DIVISIONS ON ABSCISSA,
     ¬ ORDINATE"
170 PRINT"(8 OR LESS ON ABSCISSA,
     ¬ 10 OR LESS"
180 PRINT"ON ORDINATE TO PERMIT NUMERIC ¬
     ¬LABELS)."
190 INPUTLA,LO
200 IFLA<=8ANDLO<=10THEN220
210 PRINT"↓TOO CROWDED FOR NUMERIC ¬
     ¬LABELS!↓":GOTO160
220 PRINT"              ↓↓↓rWORKING"
230 L=LA:GOSUB530
240 DU=256*C/R:REM SCALING FACTOR FOR ¬
     ¬ABSCISSA DIVISIONS.
250 AD=ABS(A-A1)*256/R
260 AA=A:BA=B:CA=C
270 CF=(B1-A1)/31:REM SCALING FACTOR ¬
     ¬FOR X
280 FOR I=1TO32:X(I)=A1+(I-1)*CF
290 Y(I)=FNA(X(I)):NEXT:REM STORE ¬
     ¬FUNCTION VALUES IN AN ARRAY
300 GOSUB430
310 DF=160/(MAX-MIN):REM SCALING FACTOR ¬
     ¬FOR F(X) ONTO 20*8=160 UNITS.
320 L=LO:A1=MIN:B1=MAX:GOSUB530
330 REM NEW INPUT FOR AUTO.SELECTION OF ¬
     ¬PLOTTING LIMITS SUBROUTINE.
340 DW=160*C/R:REM SCALING FACTOR FOR ¬
     ¬ORDINATE DIVISIONS.
350 OD=ABS(A-A1)*160/R:REM A,A1,
     ¬R ARE DIFFERENT FROM THOSE IN ¬
     ¬LINE 210
360 GOSUB970:REM PRINT COORDINATES.
370 FORI=1TO32:Y1=(Y(I)-MIN)*DF
380 Y%=Y1/8:V%=Y1-8*Y%+1
390 PRINT"h→→→→→→";D$;LEFT$(X$,I);LEFT$(
     ¬Y$,Y%);MID$(V$,V%,1);:NEXT:
     ¬REM PLOT.
400 GETZ$:IFZ$=""THEN400
410 IFZ$="F"THENPRINT"ħ":LIST120
420 PRINT"ħ":GOTO140
```

```
430 MAX=Y(1):MIN=Y(1):REM FIND LIMITS ¬
      ¬OF F(X) FOR SCALING
440 FOR I=2TO32:IFY(I)>MAXTHENMAX=Y(I):
      ¬GOTO460
450 IFMIN>Y(I)THENMIN=Y(I)
460 NEXT:RETURN
470 REM LINES 530-910 ARE THE AUTOMATIC ¬
      ¬PLOT LIMIT SELECTION PROGRAM WHICH
480 REM USES: A1,B1=RAW LOW,HIGH LIMITS
490 REM AND  L=NUMBER OF DIVISIONS/AXIS
500 REM AND RETURNS:
510 REM A,B=ROUNDED LOW,HIGH LIMITS ¬
      ¬WHICH ARE >=A1,B1, AND
520 REM C=(ROUND) INCREMENT BETWEEN ¬
      ¬DIVISIONS
530 M=12
540 FORI=1TO12:READS(I):NEXT
550 DATA1,1.5,2,2.5,3,4,5,6,7,8,9,10
560 RESTORE
570 A2=A1:B2=B1:R=B2-A2
580 C2=ABS(R/L):P=LOG(C2)/LOG(10)
590 IFP>0THEN610
600 P=P-1
610 N=SGN(P)*INT(ABS(P))
620 F=C2/10^N
630 FORJ=1TOM
640 IFF>S(J)THEN660
650 C=S(J)*(10^N):K=J:GOTO710
660 NEXTJ
670 PRINT"ABORT--MOST PROBABLE CASE IS ¬
      ¬AN ERROR
680 PRINT"IN THE TABLE OF STANDARD ¬
      ¬INCREMENT MANTISSAS
690 STOP
700 K=K+1:C=S(K)*(10^N)
710 I=A2/C
720 I=SGN(I)*INT(ABS(I)):T=I:A=T*C
730 D=ABS((A-A2)/R)
740 IFA2<0THEN800
750 IFR>0THEN840
760 IFD<=.0001THEN780
770 A=A+C
780 C=-C
790 GOTO840
800 IFR>0THEN820
810 C=-C:GOTO840
820 IFD<=.0001THEN840
830 A=A-C
840 T2=A+(C*L)
850 IFR<0THEN900
860 IFT2>B2THEN910
870 IFABS((B2-T2)/R)<=.0001THEN910
880 IFK<MTHEN700
890 N=N+1:GOTO620
900 IFT2>B2THEN870
910 B=T2:RETURN
920 REM LINES 970-1160 PRINT THE ¬
      ¬LABELED COORDINATES, USING L AND ¬
      ¬THE
930 REM QUANTITIES A,B,C RETURNED BY
940 REM THE PREVIOUS SUBROUTINE. THESE ¬
      ¬WERE OBTAINED TWICE AND THOSE FOR ¬
      ¬THE
950 REM ABSCISSA ARE THEREFORE STORED ¬
      ¬AND SUPPLIED BY THE MAIN PROGRAM
960 REM(LINES 70-420)AS LA,CA,ETC...
970 PRINT"h";:FORI=1TO20:PRINTTAB(7)"%":
      ¬NEXT
980 PRINT"h↓↓↓";TAB(4)"Y";D$;LEFT$(X$,
      ¬18);"X"
990 FORI=0TOL-1
1000 YW=I*DW-OD:YW%=YW/8:W%=YW-8*YW%+1
1010 IFYW%<0THEN1050
1020 IFA+C*I>=A2ANDA+C*I<=B2THEN1040
1030 IFA+C*I>B2THEN1060
1040 PRINT"h";D$;LEFT$(Y$,YW%);TAB(0)A+C
      ¬*I;TAB(7)MID$(V$,W%,1)
1050 NEXT
1060 PRINT"h→→→→→→→";D$;:FORI=1TO32:
      ¬PRINT"S";:NEXT
1070 FORI=0TOLA-1
1080 XU=I*DU-AD:XU%=XU/8:U%=XU-8*XU%+1
1090 IFXU%<0THEN1150
1100 IFAA+CA*I>=AXANDAA+CA*I<=BXTHEN1120
1110 IFAA+CA*I>BXTHEN1160
1120 PRINT"h→→→→→→→";D$;:IFI>0THENPRINT"
      ¬←";
1130 PRINTLEFT$(X$,XU%);MID$(U$,U%,1);
1140 IFAA+CA*I>=AXANDPOS(0)<38THENPRINT"
      ¬↓↓←←←"AA+CA*I;
1150 NEXTI
1160 RETURN
READY.
```

©

ROM Expansion For The Commodore PET

F. Arthur Cochrane
E. I. du Pont de Nemours & Co., Inc.
Savannah River Laboratory
Aiken, SC 29808

This paper was prepared in connection with work under Contract No. DE-AC09-76SR00001 with the U.S. Department of Energy.

Summary

The Commodore PET is a low-cost personal computer housed in a self-contained unit complete with keyboard and a video display screen. The PET is portable and can sit on a desk or tabletop and operates on normal household current. The PET is simply plugged in and turned on to activate its BASIC interpreter. The BASIC interpreter is a computer program that is permanently built into every PET. In contrast, a program loaded from a tape, disk, or the keyboard is placed into Read/Write memory (RAM) and is lost when the PET is turned off.

The PET has three empty sockets for expansion read-only memory units (ROMs) which have memory addresses 36864 ($9000) through 49151 ($BFFF). A method for using single-supply, erasable, programmable, read-only memories (EPROMs) with the Commodore PET microcomputer has been developed. These EPROMs will allow the development of special-purpose firmware for the PET that does not have to be reloaded each time the PET is turned on and will permit an inexperienced operator to use the PET.

Current Firmware For The Pet

There are several ROMs available for the PET:

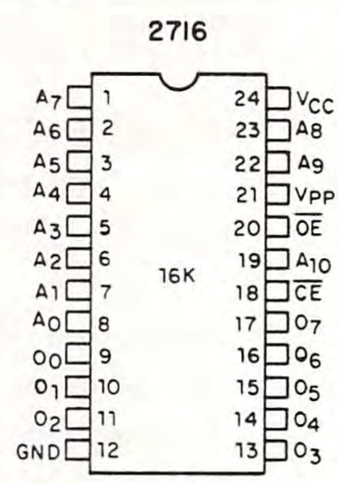
1. Commodore's WordPro-2 is a 4K ROM for addresses $B000-$BFFF.
2. Commodore's WordPro-3 is a 4K ROM for addresses $A000-$AFFF.
3. The BASIC Programmer's Toolkit™ is a 2K ROM for addresses $B000-$B7FF.
4. The Database Management System (United Software of America, New York, N.Y.) is a 2K ROM for addresses $A000-$A7FF.
5. The Keyed Random Access Method (KRAM 1.0; United Software of America) is a 2K ROM for addresses $A800-$AFFF.
6. The Monjana/1 CBM™ Monitor (Elcomp Publishing, Inc., Chino, Calif.) is a new 2K monitor ROM which is addressed for $9000-$97FF.
7. The Jinsam™ database is a 2K ROM for $9000-$97FF.

The Spacemaker™ allows two ROMs to be plugged in at the same time; a single-pole switch selects one or the other of the ROMs. The Database Management and KRAM 1.0 ROMs can thus be put on a Spacemaker. Better yet, a small wirewrap board could be built with three 24-pin sockets wired in parallel: the first socket for a ribbon cable input; the second, for the Database ROM; and the third socket for the KRAM ROM. A Spacemaker could then be used with these two ROMs and with another ROM such as the WordPro-3 ROM.

Use Of Eproms

The three empty sockets for ROM expansion are 4K sockets connected as shown in Figure 1.

There are three EPROMs, the Intel™ 2716 (2K, Figure 2), the TI™ 2516 (2K, Figure 2), and



PIN NAMES

A_0-A_{10}	ADDRESSES
CE/PGM	CHIP ENABLE/PROGRAM
OE	OUTPUT ENABLE
O_0-O_7	OUTPUTS

FIGURE 2. Pin Configuration and Pin Names of the Intel 2716 2K EPROM

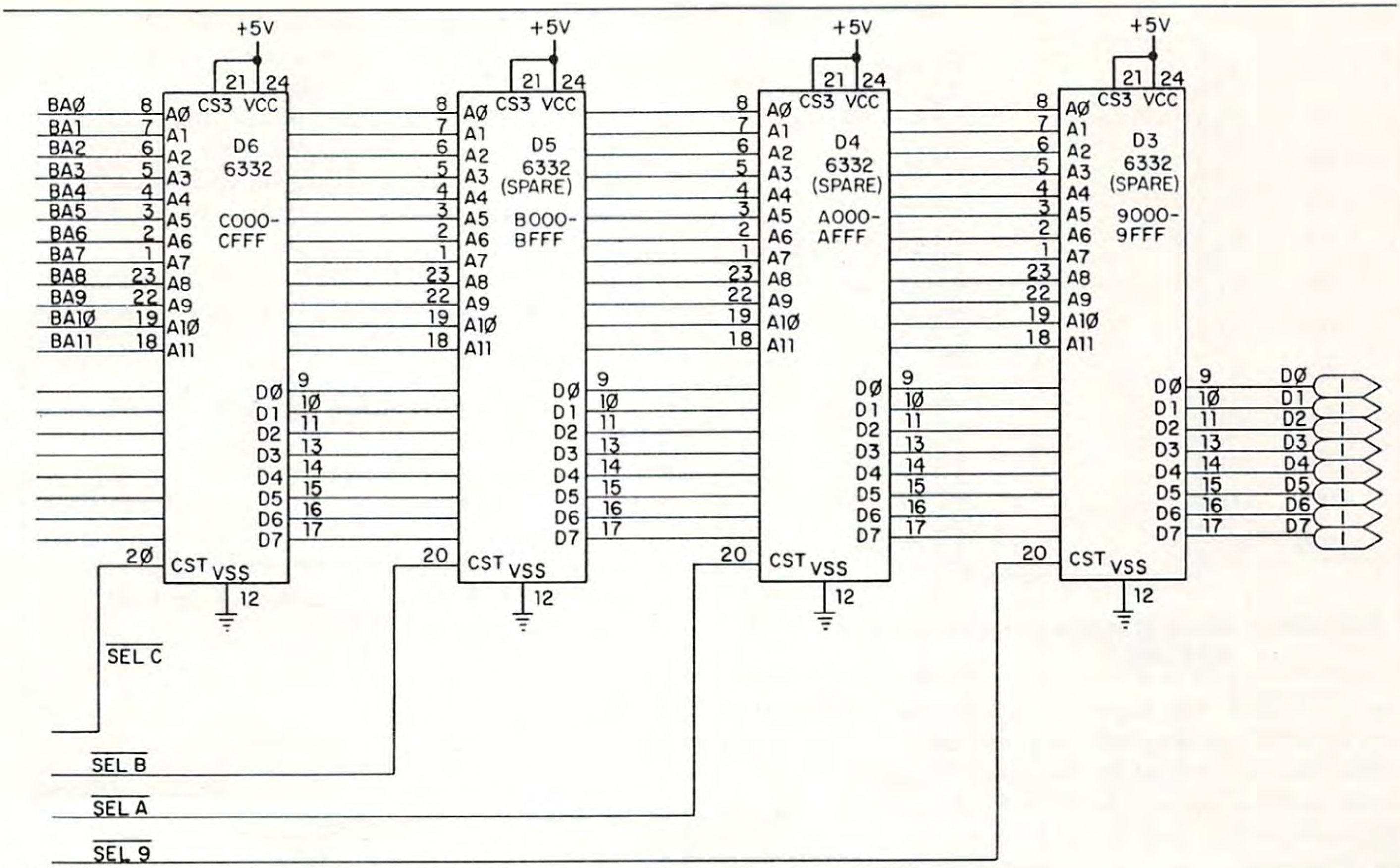


FIGURE 1. Schematic Diagram of PET ROM Modification

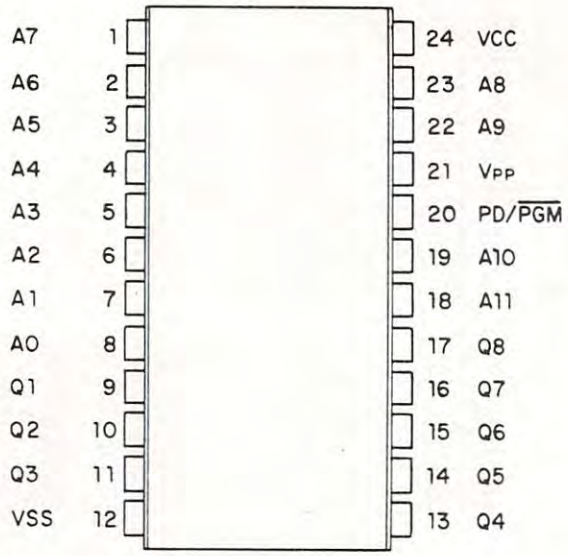


FIGURE 3. Block Diagram of the TI 2532 EPROM

the TI™ 2532 (4K, Figure 3), which can be plugged into the PET sockets with no problems. The 2K EPROMs are enabled for the lower 2K of the 4K socket address because Address Bit 11 acts as a chip select. However, these EPROMs can't be used with the Spacemaker because it uses Socket Pin 21, a high-enable chip select on 2K and 4K ROMs, to select one of its two sockets.

We have developed circuitry to overcome these difficulties and allow combinations of EPROMs with the Spacemaker. The circuits described provide a full 12K of expansion ROM space.

Intel also makes the 2732 EPROM 4K (Figure 4) which can't be used directly with either the PET or Spacemaker; however, with the switch of a wire and an inverter, the 2732 EPROM can be used with both the PET and Spacemaker. To use the 2732 EPROM, connect from Socket Pin 18 to Socket Pin 21 of the 2732, which connects All. Then go from Socket Pin 21 through an inverter to Socket Pin 18 of the 2732, which allows use with the Spacemaker. See Figure 5 for a schematic diagram.

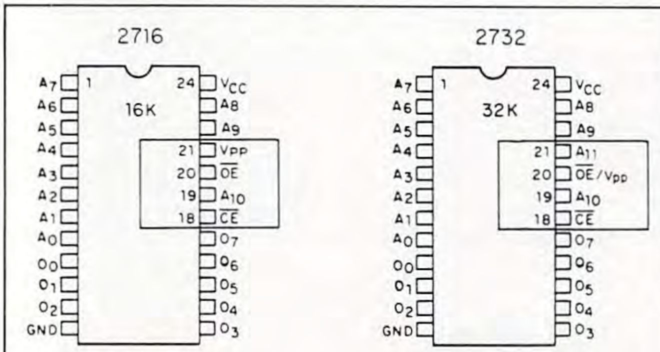


FIGURE 4. Block Diagram of the Intel 2732 EPROM

The TI 2532 eprom can be used with the Spacemaker by connecting Socket Pin 20 through an inverter to a NAND gate, connecting Socket Pin 21 to the same NAND gate, and finally, the output of the NAND gate to Socket Pin 20 of the TI 2352. Also Socket Pin 21 (Vpp) of the TI 2532 must be connected to a 5-volt supply (Socket Pin 24). See Figure 6 for a schematic diagram.

The 2716 and 2516 2K EPROMs can't be used directly with the Spacemaker, so a small board has to

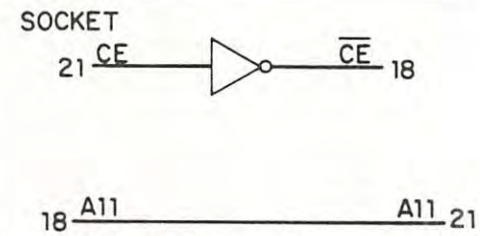


FIGURE 5. Schematic Diagram of the Intel 2732 EPROM

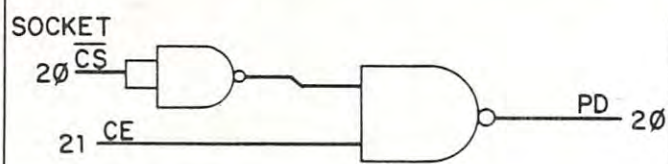


FIGURE 6. Schematic Diagram of the TI 2532 EPROM

be built that includes enough decoding to allow one 2K EPROM for the lower 2K of the 4K available from the socket, and another 2K EPROM for the upper 2K address space. A three-to-eight decoder is used by Decoding Pin 21 (the high chip select from the Spacemaker), Pin 20 (the low chip select from the PET), and Pin 18 (Address Bit 11). Pins 18 and 20 of the EPROMs are connected together (as both are chip selects) and are connected to the decoder output as given in Table 1. Figure 7 shows a PC board with (from left to right) the ribbon cable input, the BASIC Programmer's Toolkit, a 2716 ROM, and the 3-to-8 decoder.

Table 1 Truth Table for Decoder

C (21)	B (20)	A (18)	
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	Select lower 2K 2716
1	0	1	Select upper 2K 2716
1	1	0	
1	1	1	

C (21) is the high chip select from the Spacemaker.
B (20) is the low chip select from the PET.
A (18) is Address Bit 11 from the PET.

Additional Space Available

The PET has four ROMs (three 4K ROMs and one 2K ROM) for its 14K operating system. The 2K ROM is addressed from $E000 to $E7FF. The PET uses $E800 to $E8FF for the keyboard scan chip (6520), the IEEE interface (6520), and the USER PORT (6522) chips.

The address space $E900 to $EFFF is not used and is available for expansion. It can be used with a modification of the method for 2K EPROMs and a 2716 EPROM. The method is the same as that described above, except that the 2716 EPROM is not enabled for the first 256-bytes of its address space. This can be done with the 3-to-8 decoder enabling one of the chip selects and Address Lines 8, 9, and 10 through a three-input NOR gate. This enables the other chip select. Using the NOR gate will disable the EPROM for the first 256 bytes of the 2K range (Figure 8).

FIGURE 8. Schematic Diagram of the $E900 Address Space

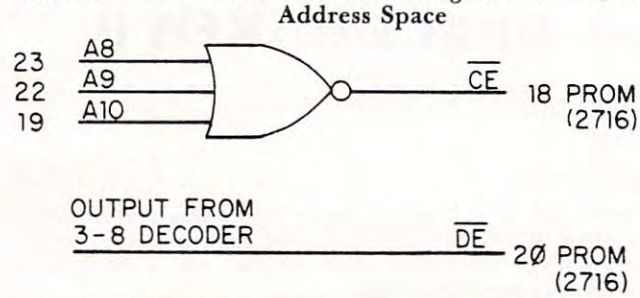


Figure 9 shows the inside of a PET with two Spacemakers (the vertical boards in the $A000 and $B000 ROM socket, and a PC board modified to use $E900-$EFFF.

All the boards mentioned in this article are connected to the PET or Spacemaker socket with a 24-pin dip socket to dip socket ribbon cable. The extra circuitry is powered from Pin 24 (5 V) and from Pin 12 (ground).

Conclusions

We have programmed an EPROM which includes the PET disk operating system (DOS), the support program (the WEDGE), a screen print routine, an expansion for the machine language monitor, and a routine for repeat keys for the PET. This EPROM is enabled into the system with one SYS command.

Acknowledgements

The following company trademarks are acknowledged:

PET and CBM - Trademarks of Commodore Business Machines, Inc., Norristown, Pa.

Toolkit - Trademark of Palo Alto ICs, a division of Nestar Systems, Palo Alto, Calif.

Jinsam - Trademark of Jini Micro-Systems, Inc., Bronx, N.Y.

Spacemaker - Trademark of CGRS Microtech, Inc., Langhorne, Pa.

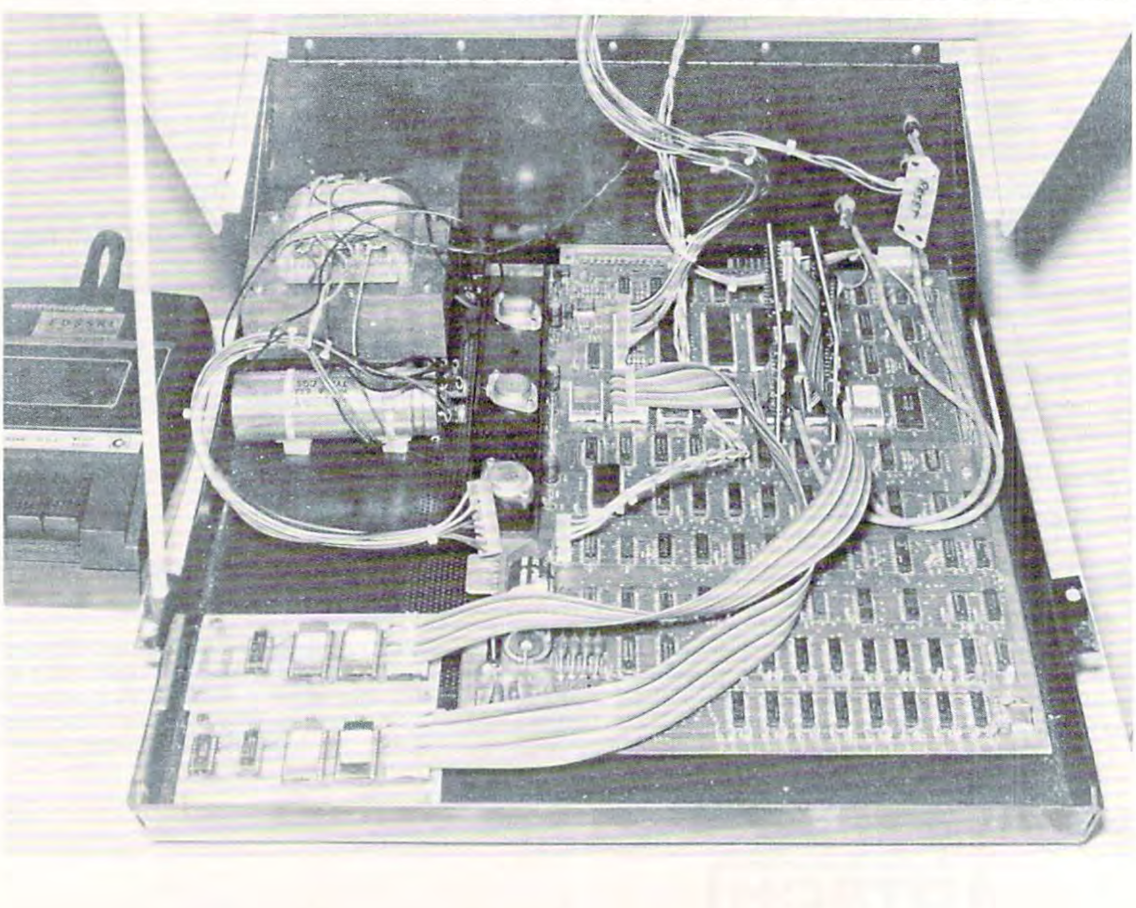
Intel - Trademark of TM Corporation, Santa Clara, Calif.

TI - Trademark of Texas Instruments, Houston, Texas.

REFERENCES

1. **Component Data Catalog,** Intel Corp., Santa Clara, California (1979).
2. **The MOS Memory Data Book,** Texas Instruments Inc., Dallas, Texas (1979).
3. **Logic Diagram Dynamic PET ROMs,** Commodore Business Machines Inc., Norristown, Pennsylvania (1980 rev.).

FIGURE 9. Interior of the Commodore PET



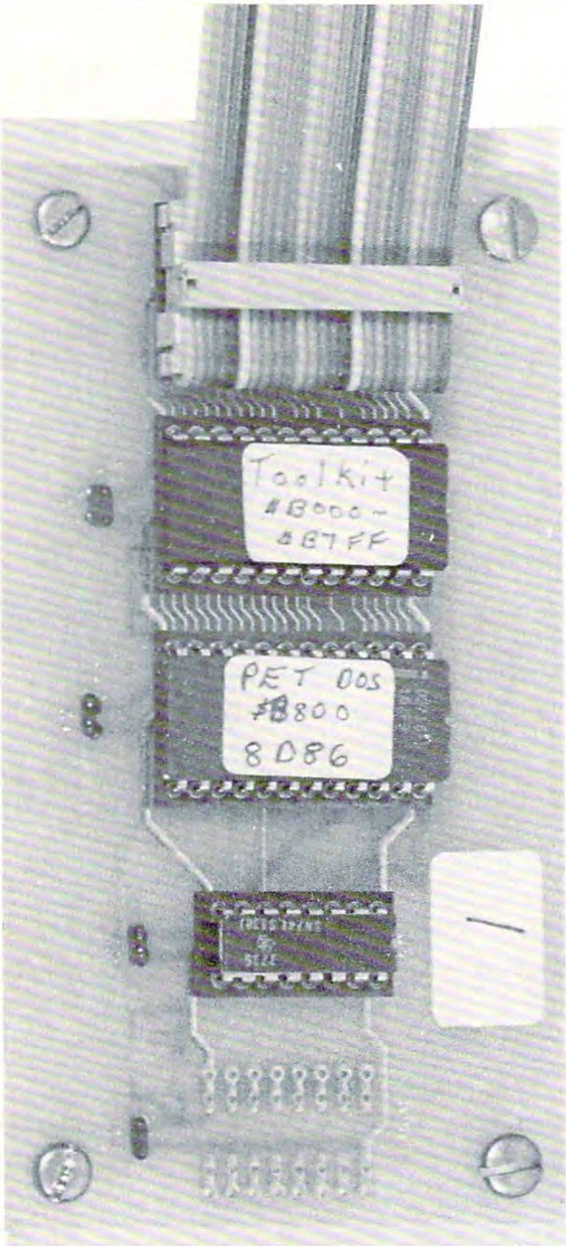


FIGURE 7. PC Board layout ©

Working With Basic 4.0

Jim Butterfield, Toronto

Machine Language Monitor

If you want to get printed output from the Machine Language Monitor in Basic 4.0, you'll have to call the monitor with SYS 54386.

Open your printer first, or course, with OPEN 4,4 and set the output with CMD4 — then give the SYS call. You can't give the usual SYS 1024 or SYS 4 commands: these get to the MLM by means of a Break command, and on the 4.0 system, a break cancels the CMD.

There's a good reason for this. The BRK (Break) instruction is normally used for debugging: it stops a program in mid-execution and allows you to examine registers and memory, and to make changes if you wish. If your program under test mistakenly set the CMD value to something unwanted, you'd lose control unless the BRK set it back. So... the Break action resets output to the screen.

When you want printed output from the MLM, however, you want to enter it without resetting the output. Use the call above, which enters the monitor directly and skips the reset part.

For normal output to the screen, the usual SYS 4 will still do the job nicely.

DOS Commands

It's nice to be able to say things like SCRATCH "WORKFILE", either as a direct statement or in a program. At first glance, however, it seems that variables won't work: you can't say SCRATCH X$, where X$ defines the file name. Similarly, you can designate drive zero within a command as D0 or drive one as D1, but you can't use DN to designate drive N, where N is a variable of value 1 or 0.

It all becomes easy when you learn the syntax. Just put the variable name in parentheses and it all works. So you can say, SCRATCH (X$), D(N) and you'll remove the file whose name is defined in X$ from the drive whose number is defined by N. This kind of syntax applies generally to the DOS commands.

IEEE-488 Timeout

Most of us will never meet a timeout on the GPIB; it only happens when certain slow devices are fitted. If you do have such a device and you get a timeout, the only solution was to go back and keep trying until the device becomes ready.

On Basic 4.0, you're allowed to defeat the timeout feature if you wish and tell the PET to wait forever. You do this by setting location 1020 (hexadecimal 03FC) to a negative value: POKE 1020, 255 will do the trick nicely in Basic. Reset the location to zero when you want to restore normal timeout action.

SOF-TEC
ANNOUNCES IT'S NEW REAL ESTATE SOFTWARE

SUPR-AN

The ultimate in User Flexibility!

Control in sensitivity analysis, Financing Terms (Any Combination), Investor Orientation. Use your own identifiers for income and expense categories, property titles, addresses, etc. Measure before and after-tax benefits, both for Future and Present Value Estimates. For any property.

Required PET/CBM 32K (specify Disk or Cassette, with or without printer)

***NOW:* $49.95** (Add 4% in Georgia)

(APPLE Version Coming May 1981!!)

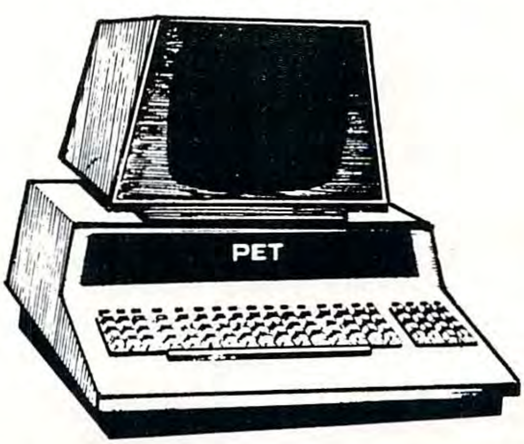
ORDER NOW

OTHER SOFTWARE AVAILABLE:

Real Estate:	Subdivision Analysis	- $49.95
	FHA 221-d(4) Analysis	- $19.95
	ELLWOOD Tables	- $14.95
Other:	Retail/Banking Market Share Analysis	- $149.95
	Merchandise Line Purchasing Power Analysis for Geographical Area	- (Call)
	Demographical Data any Area	- (Call)

ORDER NOW!!

SOF-TEC • 1043 First Ave., Columbus, Ga 31901
(404) 327-1221



Every PET™ Needs a Friend.

CURSOR is the best friend your Commodore PET will ever have. Since July, 1978 we have published 150 of the most user-friendly programs for the PET available anywhere. When we write or edit a program, we spend lots of time fussing about how it will treat you. We pay attention to lots of little things that help make using a computer a pleasure instead of a pain.

Naturally, **CURSOR** programs are technically excellent. Each program that we purchase is extensively edited or re-written by a professional programmer. But *imagination* is just as important as being user-friendly and technically good! We delight in bringing you off-beat, unusual programs that "show off" the abilities of your PET or CBM.

CURSOR is user-friendly, technically great and full of imaginative programs. And every issue of **CURSOR** is still available! We continue to upgrade previously published programs so that they'll work on the three varieties of Commodore ROM's (Old, New, and 4.0). New issues also work on the 80 column CBM.

For only $4.95 you can buy a sample issue and judge for yourself. Or send $27 for a six-issue subscription. Each **CURSOR** comes to you as a C-30 cassette with five programs and a graphic Front Cover, ready to LOAD and RUN on your PET.

Who knows? After your PET meets **CURSOR**, things may never be the same!

Published By:

THE CODE WORKS
Box 550
Goleta, CA 93116
805-683-1585

Distributed by:
AUDIOGENIC Ltd.
P. O. Box 88
Reading, Berkshire

SYSTEMS FORMULATE Corp.
Shin-Makicho Bldg., 1-8-17
Yaesu, Chuo-Ku, Tokyo 103

Review:

Papermate Wordprocessor

Paul W. Sparks
Alexandria, Va.

This is a review of PAPERMATE, developed by Michael Riley and sold by A B Computers for $29.95. (115 E. Stump Road, Montgomeryville, PA 18936).

The main reason that I became interested in this software is that I have an 8K PET with upgraded memory to 32K, both Commodore and Computhink disk units and both a Commodore and a non-standard printer. With a system like this, other available word processors are not as attractive because they cannot be used with the entire system and may require extra ROM boards to install. However, after I received this system I think that even users with a more conventional setup will be interested in using PAPERMATE. It can be used with old and new ROMs, is written in BASIC and ML, and can use either a CBM or an ASCII printer. It also uses a CBM disk or can be user modified to use another disk unit. It has variable line spacing within text, variable margins within text, shift for upper case in either ROM set, all caps look, auto repeat for all keys, center text within text, right justify, multiple tab control, edit, delete, insert, block handling and out of text, header, footer, page numbering, scrolling either direction, and most words on the screen shift to avoid splitting. And it's fast.

Let's review the PAPERMATE commands. First of all there are program commands and in-text commands. The program commands are given by exiting the text to a list of one letter entries to perform functions including block handling and disk operations. The in-text commands can be added or changed at any time and are primarily printer commands such as tabs and justification.

Program Commands:

SAVE: This command allows a text to be saved on disk. It will allow for either CBM disk or tape. Other units may be used if programmed by the user.

LOAD: This command allows loading a text from disk or tape.

WRITE: This command puts the user into the mode to enter text from the keyboard. The text may be edited at all times and the text is always available unless it is replaced by a LOAD which will put another text in memory. In-text commands may be added as you go along.

PRINT: This command will print out your text on the printer of your choice. All in-text commands will be translated into printer variables. You have the ability to stop the printout at any time and there is a feature to pause the printout at the end of each page so that if you are using single sheets they may be replaced.

FORMAT: This feature will cleanup the screen presentation if you have been inserting or deleting a lot of single words or letters in text.

KEY DEFINITION: This will allow you to change the output of one or more keys of your choice to whatever you want. The STOP key is already predefined to give a "." while in the WRITE mode. If you don't like your keyboard you can "change" it.

COMMAND: This feature allows you to give your CBM disk any command that would be used with a PRINT#15 (command channel mode).

The following commands are all block commands. The block is defined in the text by using the bracket characters to define the beginning and the end of the block.

TRANSFER: This command will transfer the defined block to be inserted at the point that the cursor was left (in the margin) before going to the program command section.

DELETE: This command will delete all text that is included in the block.

MEMORIZE: This command will allow the user to place on disk the defined block of text (to be used later).

APPEND: This command will add any block of text to the end of the file presently in memory. This block can then be moved anywhere desired using the TRANSFER command.

INSERT: This command will insert blanks equal to the size of the defined block and move the text down out of the way. It makes space.

In-Text Commands:

MARGIN: The left and right margins may be set and changed at any time. A command is available to decrease the left margin setting by so many spaces.

LINE CONTROL: Line spacing (double, triple, and more) can be controlled and changed in text. The program will allow for the option to right justify. There is a command to center the next (number supplied by the user) several lines of text. There is also a semi-automatic mode to use hyphens. One of the best features is the ability to set up multiple TABs and to change them at will.

Professional Business Software

For The Commodore 32K Microcomputer System With 2040 Dual Drive Disk & 2022 Tractor Feed Printer

General Ledger

- Holds Up To 300 Accounts.
- Accepts Up To 3000 Transactions Per Month.
- Cash Disbursements Journal, Cash Receipts Journal, and Petty Cash Journal for simplified data entry.
- Maintains Account Balances For Present Month, Present Quarter, Present Year, Three Previous Quarters, And Previous Year.
- Complete Financial Reports Including Trial Balance, Balance Sheet, Profit & Loss Statement, Cash Receipts Journal, Cash Disbursements Journal, Petty Cash Journal and more.
- Accepts Postings From External Sources Such As Accounts Payable, Accounts Receivable, Payroll, Etc. . . . .

Accounts Payable

- Interactive Data Entry With Verified Input And Complete Operator Prompting.
- Automatic Application Of Credit And Debit Memos.
- Maintains Complete Purchase Records For Up To 200 Vendors.
- Invoice File Accepts Up To 400 Invoices.
- Random Access File Organization Allows Fast Individual Record Updating.
- Multiple Reports Provide A Complete Audit Trail.
- Check Printing With Full Invoice Detail.
- Full Invoice Aging.
- Automatic Posting To General Ledger . . . .

Accounts Receivable

- Maintains Invoice File For Up To 300 Invoices.
- Accomodates Full Or Partial Invoice Payments.
- Customer File Maintains Purchase Information For Up To 1000 Customers.
- Allows For Automatic Progress Billing.
- Provides For Credit And Debit Memos As Well As Invoices.
- Prints Individualized Customer Statements.
- Interactive Data Entry With FullOperator Prompting.
- Complete Data Input Verification And Formating.
- Automatic Posting To General Ledger . . . .

Payroll

- Maintains Monthly, Quarterly, And Yearly Cumulative Totals For Each Employee.
- Payroll Check Printing With Full Deduction And Pay Detail.
- Sixteen Different Reports Including W2 And 941.
- Interactive Data Entry With Easy Correction Of Entry Errors.
- Automatic Data Verification.
- Complete Job Costing Option With Cumulative Totals And Overhead Calculations.
- Random Access File Organization For Fast Updating Of Individual Records.
- Automatic Posting To General Ledger . . . .

Structured around the time tested and reliability proven series of business software systems developed by Osbome and Associates, these programs have been designed to fill the need of a comprehensive accounting package for the new Commodore PET micro computer system. Each program can either stand alone, or be integrated with the others in a total software system.

Designed with the first time user in mind, these programs lead the operator through step by step, verified data entry. It is impossible to 'crash' a program due to operator error or invalid data input. Design consistency has been maintained from program to program to greatly increase operator familiarity and confidence.

Documentation, normally a problem for small systems users, is provided by the comprehensive series of Osborne and Associates user manuals. These three manuals together total over 800 pages of detailed step by step instructions written at three levels for DP Department Managers, Data Entry Operators, and Programmers. You don't have to worry about getting 'promises' instead of documentation because the documentation was written before the programs were developed. A second set of manuals details any changes required during conversion. Each program provided on disk with complete documentation. Packaged in a handsome three ring binder with pockets and twelve monthly dividers for convenient storage of reports.

See your nearest Commodore dealer for a demonstration of this outstanding business software system.

CMS Software Systems

5115 MENEFEE DRIVE • DALLAS, TX 75227 • 214-381-0690

PAGING: The page size may be specified (normally 66 lines). A header to be printed at the top of each page and a page number at the bottom of each page can be used. Two other paging commands are a pause feature so that paper may be changed at the end of each document and a command that will force the printer to go to the end of the page if a given number of lines are not available. This is very useful with tables or figures so that they are printed without being broken up between two pages.

LINKING: A title for each file is used. This will automatically be setup for disk use. A command that will call out the next file to be linked in a printing operation is available and any number of files may be joined in this manner.

FORM DOCUMENTS: There are several commands that allow the user to setup a form letter or any other type of recurring document and use a mailing list, etc. to fill the required information. The files may be set up using PAPERMATE or a separate program may be used. The file requirements are delineated.

GRAPHICS: There are two special commands that allow control characters to be sent to the printer. Therefore the secondary address on the CBM printer may be changed and enhanced graphics may be utilized.

MISC: There is a command that allows the user to insert non-printable remarks in the text.

There are a few non-command features that should be noted. All keys have a simple repeat feature that is activated by holding the key down for more than a second. The system will coexist with both disk DOSs and old and new BASIC. I do not know how it will work with BASIC 4.0 *(Editor's Note: We checked with the vendor, and Papermate runs on 4.0 and 80 column machines. RCL)* A simple method is described to convert for an 80 column CBM so that would indicate that it can be used with the new operating system. I would want to verify that before I bought it for that purpose. I have used it with both versions of the disk operating system successfully. One other neat little function is that if you type the REV key then the text will be typed in upper case for letter characters only and numbers, punctuation, etc. will be printed normally.

So what is wrong with it? As with many software packages I have seen, the documentation leaves a little bit to be desired. Some of the pages did not print very well. Many of the explanations were written by and for someone already familiar with the system and although they make good sense now they were quite hard to figure out at first. I believe that this system is equivalent to WORDPRO II. One must keep track of what file ties into where if you are going to insert blocks here and there. That is a function that only the much more expensive systems will perform but PAPERMATE will nicely chain files together for printing just by a single command at the end of each file. One last potential problem; if you are a novice programmer you will have considerable difficulty following Michael Riley's programming. He has done an excellent job of utilizing some very unique skills to get a lot of power from a comparatively short program and it is not easy to follow. I have changed several areas for my own idiosyncrasies and even though there is a list of variables, and program sections are defined, it was not easy and there are many pitfalls.

To summarize, the best recommendation that I can give is that this is the first program that I have been able to get my wife to use in the three years that I have owned my PET. You ought to consider this alternative if you are thinking of getting a word-processor program for your PET/CBM. ©

Disabling the STOP key

Jim Butterfield, Toronto

The STOP key can be disabled so that a program cannot be accidentally (or deliberately) stopped.

METHOD A is quick. Any cassette tape activity will reset the STOP key to its normal function, however.

```
Original ROM:  Disable STOP with POKE 537,136
               Restore STOP with POKE 537,133
Upgrade ROM:   Disable STOP with POKE 144,49
               Restore STOP with POKE 144,46
4.0 ROM:       Disable STOP with POKE 144,88
               Restore STOP with POKE 144,85
```

Method A disconnects the computer's clock (TI and TI$). If your program needs these, use method B.

METHOD B is more lengthy, but does not disturb the clocks. This method prohibits cassette tape activity.

```
Original ROM:  100 R$ = "20>:??:9??8 = 09024<88
                 >6"
               110 FOR I = 1 TO LEN(R$)/2
               120 POKE I + 900,ASC(MID$(R$,
                 I*2- - 1))*16 + ASC(MID$(R$,
                 I*2)) - 816 : NEXT I
      After the above has run:
               Disable STOP with POKE 538,3
               Restore STOP with POKE 538,230
Upgrade ROM:   100 R$ = "20>:??:9??8 = 9;004<31
                 >6"
               110 FOR I = 1 TO LEN(R$)/2
               120 POKE I + 813,ASC(MID$(R$,I*
                 2 - 1))*16 + ASC(MID$(R$,I*2)) -
                 816 : NEXT I
      After the above has run:
               Disable STOP with POKE 145,3
               Restore STOP with POKE 145,230
4.0 ROM:       100 R$ = "20>:??:9??8 = 9;004<58
                 >4"
               110 FOR I<1 TO LEN(R$)/2
               120 POKE I + 852,ASC(MID$(R$,I*
                 2 - 1))*16 + ASC(MID$*R$,I*2)) -
                 816 : NEXT I
      After the above has run:
               Disable STOP with POKE 145,3
               Restore STOP with POKE 145,228
```

How they work: Method A skips the clock update and stop key test. Method B builds a small program into low memory which allows the clock update and stop key test to be performed, but then nullifies the result of this test. The small program for method B is contained in R$ in "pig hexadecimal" format. Machine language programmers would read this as: 20 EA FF (do clock update, stop key test) A9 FF 8D 9B 00 (cancel stop test result) 4C 58 E4 (continue with keyboard service, etc.) ©

Dissecting C. W. Moser's ASSM/TED 1.0

Francis Turco

Carl Moser's excellent assembler/text editor for the 6502 Microprocessor has been reviewed superficially in several publications.[1,2] So far, no one has done an in-depth write-up for a PET owner who wants to understand or modify his copy. The manual provided by Moser is adequate, but sketchy in some areas. I, for one, would like to see some articles by users who have figured out solutions to problem areas.

For example, PET owners find out (on page 35 of the manual) that "At present, the ASSM/TED does not contain a printer subroutine...". In another area, the ASSM/TED is designed for a "standard" PET and utilizes the audio cassette drives for off-line storage. The manual (Sections 6 & 7) discusses configuring the ASSM/TED for disk operation and using it with disk. This discussion is too brief to be understandable by a novice assembly language programmer.

In still another area, the editor has many powerful capabilities and will accept a full line of characters (65 typed characters) but the sense of the shift key is reversed. That is, shift gives lower case letters. Unshifted gives upper case letters. This proves to be cumbersome when typing a letter or manuscript from the PET keyboard.

In an effort to shed some light for others, who like myself, are trying to understand and modify their copy of ASSM/TED and perhaps stimulate some of you to share your findings, I am submitting some areas that I have uncovered in Moser's Assembler.

Figure 1 shows a memory map of the assembler/text editor. The assembler is written for a 16K PET and fills almost all useable memory space. As the figure shows, the assembler and text editor are co-resident and occupy the space from $2000 thru $3FFF. Commodore's monitor occupies the area from $0400 thru $076C. This leaves enough memory for a relocatable file ($1F00 thru $1FFF), a lable file ($1800 thru $1EFF), and approximately 4K for user programs ($0770 thru $17FF).

Table I is a list of addresses of major routines. This is a fun table -- try some experimenting with it. For example, RUN 8390 will assemble your program. RUN 8390 LIST will assemble and list. RUN 8470 will print your program. Table II provides a list of addresses of the pseudo opcode routines, while Table III contains some interesting areas that will be helpful to someone modifying his assembler.

Carl Moser's ASSM/TED is a very good program and will allow the PET owner to convert his PET into a 6502 development station with a little effort on his part. If the PET is equipped with a line printer off the IEEE port, the owner can easily get around the first problem area and get a listing of his source code and/or his assembly. This subject will be treated in PART II of this article.

1. **Compute**, Fall 1979, p. 100, "6502 Macro Assembler and Text Editor SYM Version" by Harvey Herman
2. **The PAPER**, Vol. II, Issue 6, August 1979, "Relocating Macro Assembler/Text Editor 1.0 by R. Busdieker

Figure 1. ASSM/TED 1.0 Memory Map

HEX	DEC	
3FFF 2000	16383 8192	ASSEMBLER & TEXT EDITOR by C. W. MOSER
1FFF 1F00	8191 7936	RELOCATABLE FILE (256 BYTE BUFFER)
1EFF 1800	7935 6144	LABEL FILE (SYMBOL TABLE)
17FF 0770	6143 1904	USER'S TEXT FILE (SOURCE CODE)
076C 0400	1900 1024	COMMODORE'S MONITOR (876 BYTES)
03FF 0000	1023 0	RESERVED FOR COMMODORE'S OPERATING SYSTEM

Table I
MAJOR ASSEMBLER ROUTINES

HEX	DEC	ROUTINE	
2033	8243	CLEAR	user's text file
208A	8330	BREAK	to monitor
2098	8344	AUTO	line number
20A0	8352	GET	program from tape
20A6	8358	FORMAT	text file
20B6	8374	MANUSCRIPT	line numbers output/ not output
20C6	8390	ASSEMBLE	source code
20FF	8447	RUN	program previously assembled
2116	8470	PRINT	text file
2AFB	11003	OUTPUT	create a relocatable object file
2E52	11858	LABELS	prints out label file
31EE	12782	PASS	execute the second pass of assembly
333E	13118	NUMBER	re-number text file
3467	13415	PUT	program out to tape
3559	13657	FIND	character string specified
355F	13663	EDIT	change source code
3844	14404	HARD	print routine (not functional on PET)
3873	14451	COPY	lines of text
39B9	14777	MOVE	lines of text

39C2	14786	DELETE	lines of text
39EF	14831	SET	boundaries of text file, label file & buffer
3A80	14976	DUPLICATE	files from tape 1 to tape 0
3AB6	15030	ENTER	file name in the diskette directory
3AC7	15047	LOOK UP	file name in the diskette directory
3B50	15184	SHIFT	upper/lower case

Table II
PSEUDO OPCODE ROUTINES

HEX	DEC	ROUTINE	
2919	10521	.DS	Designate Storage
2964	10596	.EJ	Eject
297B	10619	.RS	Resolve address & Store
2980	10624	.CE	Continue with Errors
2985	10629	.OS	Object Store option
298A	10634	.OC	Object store option Clear
298F	10639	.CT	Continues on Tape
2994	10644	.LS	List option Set
2999	10649	.LC	List Option Clear
299F	10655	.SI	Store Internal address
29A8	10664	.SE	Store External address
29B3	10675	.BA	Beginning Address
29F3	10739	.MC	Move Code
2A1D	10781	.BY	Bytes
2A57	10839	.DI	Designate Internal
2A60	10848	.DE	Designate External
2AB7	10935	.EN	End
3378	13176	.RC	Resolve Code
3D1E	15646	.ES	Output macro generated object code
3D23	15651	.EC	Supress macro generated object code
3D6A	15722	.MD	Macro Definition
3E0C	15884	.ME	Macro End

Table III
INTERESTING AREAS

HEX	DEC	ROUTINE
2000	8192	Cold start of ASSM/TED 1.0
203F	8255	Command Line Interpreter
207A	8314	Initializes Pointer for Text File
2090	8336	Warm start of ASSM/TED 1.0
2190	8592	Same as 8599 + carriage return
2197	8599	Prints out the double slash after listing
2602	9730	Reads remainder of entered command - For Example: PRINT 100 200 or FORMAT CLEAR
26AB	9899	Jump Table for Major Assembler Routines (Commands)
271C	10012	Pseudo Opcode Table
27AA	10154	Mnemonics Table
2E89	11913	Xfers Pointer for Lable File to Zero Page Initialize Pointer for Lable File
2F96	12182	Stores a Zero Pointer +2
32DB	13019	Prints character that is in accumulator (same function as 65490 in BASIC ROM)
330B	13067	Prints carriage return
331A	13082	Prints 2 spaces
331D	13085	Prints 1 space
3323	13091	Converts accumulator to Hex & prints it
354F-3558	13647-13656	Permanent Copy of Value of Boundaries for Text, Label & Buffer (See also 14889)
37E2	14306	Moser suggests this location for a JSR to a line printer routine written by the user. The routine at 13019 would call this subroutine.
3A29	14889	Prints out the boundaries & the present end of data (See also 13647)
3F00-3FFF	16128-16383	Relocated Page 1 variables
3F35-3F85	16181-16261	Keyboard Buffer

©

Pet File I/O In Machine Language

Raymond A. Diedrichs

The PET's I/O scheme is very flexible. A BASIC language program can effortlessly specify that the PET read data from an input device (tape, disk, or keyboard) and send processed data to an output device (printer, plotter, or CRT). At all times the PET knows which devices are for input, which are for output, and when and how each should be used.

But there are occasions when we need an assembly language program. Must we give up the PET's facile I/O because we choose to use machine language? No, indeed. It is as easy to perform standard PET I/O in machine language as it is in BASIC, and this article will explore the necessary techniques.

Let's first be certain that we understand exactly what it is that we specify when we write BASIC language I/O statements. Consider this simple-minded program:

```
10 REM EXAMPLE ONE
20 INPUT A
30 PRINT A
```

The I/O actions ("INPUT" and "PRINT") are indicated, but the I/O devices are not. The PET therefore uses the default devices: the keyboard for input in line 20 and the CRT for output in line 30.

A second example uses explicit device indication:

```
10 REM EXAMPLE TWO
20 OPEN 1,2,0,"SAMPLE"
30 INPUT #1,A
40 INPUT B
50 PRINT A,B
```

In line 20 logical file 1 is created for input of file "SAMPLE" from cassette unit 2. In line 30 file 1

("#") is the input source, so the PET seeks input from cassette 2, which is the device associated with the file. In line 40 the PET once again uses the keyboard for input, and in line 50 the output is sent to the CRT.

There are multiple I/O devices in example two, but the PET copes by temporarily ignoring all devices except the pair which are in current use for input and output. If a file number is attached to the I/O statement, the file device is made current; if no file is indicated, the PET reverts to the default device.

When we use machine language, the default input and output devices remain the same as in BASIC. Therefore, when

```
JSR $FFCF
```

is executed, the next character in the input stream is returned in register A. Likewise, when

```
LDA #ASCII-CHARACTER
JSR $FFD2
```

is executed, the character in register A is displayed on the CRT in the next print position.

To use devices other than the default in machine language, we must open a logical file for each device and supply the same information that is supplied by the equivalent BASIC OPEN statement. The following routine performs this function:

```
!
! MACHINE LANGUAGE OPEN STATEMENT
    [NEW ROM PET]
!
    LDA #FILE-NUMBER
    STA $D2
!
! IF THE DEVICE HAS A BUFFER, DECLARE THE
    BUFFER START ADDRESS
! IF NO BUFFER IS NEEDED, SKIP THIS SECTION
!
    LDA #.LOW.ADDRESS-OF-DEVICE-BUFFER
    STA $D6
    LDA #.HI.ADDRESS OF DEVICE BUFFER
    STA $D7
!
! IF THE FILE HAS A NAME, THE NAME MUST
    RESIDE SOMEWHERE IN MEMORY
! AS AN ASCII CHARACTER STRING. DECLARE THE
    LENGTH OF THE NAME IN
! CHARACTERS AND THE STARTING MEMORY
    LOCATION OF THE NAME STRING.
! IF THE FILE DOESN'T HAVE A NAME, DECLARE
    A NAME LENGTH OF 0
    LDA #FILENAME-CHARACTER-STRING-LENGTH
    STA $D1
    LDA #.LOW.ADDRESS-OF-FILENAME-
      CHARACTER-STRING
    STA $DA
    LDA #.HI.ADDRESS-OF-FILENAME-CHARACTER-
      STRING
    STA $DB
!
! CALL A PET SYSTEM SUBROUTINE WHICH OPENS
    FILE USING THE
! INITIALIZED SYSTEM VARIABLE SET
!
    JSR $F524
```

This set of system variables can assume any values which make sense in the equivalent BASIC OPEN statement. The secondary address for a cassette file, for example, can be 0 (for READ), 1 for (WRITE), or 2 (for WRITE with EOT).

When input or output is required from a file device, we must make that device a current I/O device:

```
!
! ROUTINE TO MAKE A DEVICE THE CURRENT
    INPUT DEVICE [NEW ROM PET]
!
    LDX #FILE-NUMBER
    JSR $FFC6
!
! ROUTINE TO MAKE A DEVICE THE CURRENT
    OUTPUT DEVICE [NEW ROM PET]
!
    LDX #FILE-NUMBER
    JSR $FFC9
```

If, for example, a printer is attached in this manner, then the output routine at $FFD2 sends output directly to the printer on the IEEE-488 bus; all 'LISTEN' and 'UNLISTEN' commands are included. When the cassette is attached for input or output, the PET performs all cassette motor control, data buffering, prompting ("PRESS PLAY AND RECORD"), and logging ("SEARCHING FOR SAMPLE").

When the attached device is no longer needed (or temporarily not needed — I/O can be mixed just as example two showed), restore the default devices in this manner:

```
JSR $FFCC
```

The file device is now detached, and the file can be closed:

```
!
! ROUTINE TO CLOSE A LOGICAL FILE [NEW ROM
    PET]
!
    LDA #FILE-NUMBER
    JSR $F2AE
```

In this article, we have seen that standard PET I/O in machine language is simple, uniform, and economical of user program space. ©

How To Get Started In Machine Code

And Not Go Crazy With A Routine For Two Joysticks

Elizabeth Deal

This article is for people who are taking their first steps in machine code. The ideas are illustrated by a relocatable subroutine for two joysticks that are attached to the Pet exactly as suggested by Harvey Herman in **COMPUTE!**, vol. 1, #4, p. 89. It is written for a new-ROM Pet, with untested modifications for the old-ROM Pet.

My goal for the joystick routine was quick interpretation of joystick positions into numeric keypad equivalents of those positions. The details of conversion have been described by Harvey Herman. Location 59471 ($E84F) contains 255 ($FF) when joysticks haven't moved or another 8-bit number when they have. Four bits belong to each joystick. A Basic routine that examines the 8-bit number and converts it to a numeric keypad equivalent takes a long time. The process is instantaneous in machine code. I've added a feature of wait option. Some games require motion to occur before continuing while others do not. Instead of having to write or omit an equivalent Basic statement the wait option is entered into the program. Finally, I wanted this routine to be relocatable and coexistent with other machine code routines I have.

The January 1980 issue of **COMPUTE!** contained a plea from Mr. Schiller to make assembly listings more accessible to users. I share Mr. Schiller's concern. Note that the three different looking programs are all the same. The disassembled listing in figure 1 begins at hex location $7000 (28672 decimal). The double letter machine code sits between the address ($7000) and the assembler code (LDA © $34). The code on the first line consists of A9 34. Monitor listing in figure 2 also begins with A9 34 and so do the DATA lines in figure 3 where that pair of machine instructions has been changed to decimal 169 52.

How you feed your Pet is up to you. The quickest and most error prone way is the DATA lines. This method, however, facilitates moving code from one place to another. Entry via the Monitor is the easiest, but without a relocating program the code can't move. Follow the excellent instructions of Charles Brannon in his KEYPRINT routine (**COMPUTE!**, Nov.-Dec. 1980) to enter the code via the monitor. Save 69 bytes (program and table + 1).

Where to put the code is also up to you. The listing is at $7000. The code does not have to be at $7000. You may put it in the second cassette buffer (decimal 826 or hex $033A), behind your Basic program, at the top of your Pet or any other reasonable place. Watch out - the program flags the change in top of the Pet pointer by POKEing 894 with 1. This is to prevent Pet from losing its memory during successive corrections and reruns. Should you need to make a correction, reset the pointers to their original value and POKE 894 with any non-one value. Final caution — because of the way I implemented relocatability the subroutine must be entered by a SYS call. Independently of location and manner of entry of this code, you must tell Basic where the code is located by setting variable AD properly. My Basic program takes care of it, otherwise you're on your own.

The routine ends at $7033. Locations $7034 to $7043 contain a lookup table of key numbers corresponding to joystick motions. This table is referenced by the address in $11 and $12. Numeric keypad equivalents are returned to Basic in locations 824 and 825 ($0338 and $0339, at the tail end of the first cassette buffer).

The logic of the joystick routine will not tax your intelligence, thus you can spend all of it on learning some simple machine code. This is exactly what I did and will now describe some of my experiences.

Machine language coding is rough when you jump from a higher level language. You seem to chase your own tail going in circles for need of an instruction that will do what you want it to do. Little tools I took for granted in Basic, like addition or handling arrays become a pain when they have to be considered with the same level of seriousness as the payroll program. And if this doesn't send you to the looney bin, the counters that count to 256 and reset to zero when you least expect it and the two byte addresses coexisting with one byte accumulator will get you for sure. You will not find unconditional GOTO in the repertoire, you will not find subscripted variables (arrays), and you'll have to learn to count **backwards** in hex...

The fact is all these tools **are** in the language. You have to find them and improvise, and that's the fun part. What comes out is a different matter. When I first wrote this joystick routine, it worked but I had a nasty feeling that it was too big and complicated for such an undemanding task.

I got the nerve to contact Jim Butterfield and it was a very rewarding experience. Jim helped me come to grips with instructions. He pointed out the sloppy code and gave suggestions for making it better. He also gave me SUPERMON, which contains an assembler, disassembler and all sorts of other

PET' MACHINE LANGUAGE GUIDE

Contents include sections on:

- Input and output routines.
- Fixed point, floating point, and Ascii number conversion.
- Clocks and timers.
- Built-in arithmetic functions.
- Programming hints and suggestions.
- Many sample programs.

While supply lasts:
Guides for Old ROMS
only $5.00 inc. postage
New ROMS order below

If you are interested in or are already into machine language programming on the PET, then this invaluable guide is for you. More than 30 of the PET's built-in routines are fully detailed so that the reader can immediately put them to good use.

Available for $6.95 + .75 postage. Michigan residents please include 4% state sales tax. VISA and Mastercharge cards accepted - give card number and expiration date. Quantity discounts are available.

ABACUS SOFTWARE
P. O. Box 7211
Grand Rapids, Michigan 49510

IEEE-488 BUS
SYSTEM BUILDING BLOCKS
For Commodore PET/CBM and other computers...

TNW-2000

TNW-1000 Serial Interface: $129
1 channel output only

TNW-2000 Serial Interface: $229
1 channel input and output

TNW-232D Dual Serial Interface: $369
2 channels input and output plus RS-232 control lines

TNW-103 Telephone Modem: $389
Auto answer auto dial Use with DAA

SOFTWARE

PTERM: A program that turns your PET into a terminal (Use with TNW-2000, TNW-232D, or TNW 103)
SWAP: Allows storage of up to 8 programs in PET memory at once Run them in any order
PAN: A sophisticated electronic mail program (use with TNW-103)

PLUS Most popular computers disks, printers, etc

TNW CORPORATION

Write or call for information today:

TNW Corporation
3351 Hancock Street
San Diego CA 92110
(714) 225-1040

commodore

CANADA's Largest Commodore Dealer

We stock the full line of
PET and CBM products and accessories.

OFF THE SHELF DELIVERY

SOFTWARE — HARDWARE — SERVICE
Best Book Selection In Town
BEST BOOK SELECTION IN TOWN

Dust Covers: Pet Cover $11.95
Floppy Cover 9.95
Printer Cover (old) 9.95
Printer Cover (new) 9.95

Trade-in Your Old PET For The Latests Models

HOUSE OF COMPUTERS inc.
368 EGLINTON AVE. W. (at Avenue Rd.)
TORONTO, ONTARIO M5N 1A2
(416) 482-4336

SELECT — A — ROM
For the Commodore PET/CBM

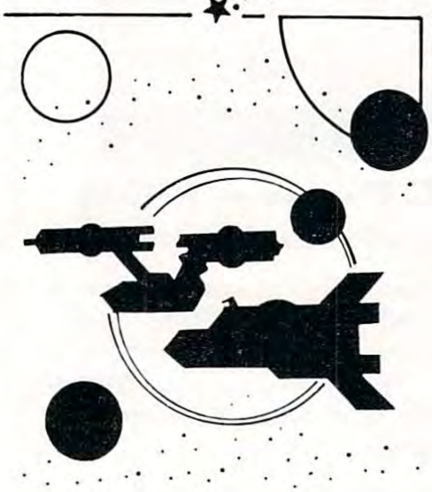
With one rotary switch select 1 to 6 separate ROMS or EPROMS, without damaging your computer board or rom pins. Now you can use **Wordpro, Toolkit, Visicalc, Eproms** as **2716's** or **2532's** etc.

Assembled $80 post paid.
Kit $45.00 post paid.

A switch between old basic 2.0 and new 4.0 basic ROMS. Now you can utilize your computer with new and old software. (Write for details).

MRJ 7951 No. 4 Rd., Richmond, B.C., Canada, V6Y 2T4
Telephone (604) 273-3651

Get Fireworks From Your PET*!



TREK-X

TREK-X Command the Enterprise as you scour the quadrant for enemy warships. This package not only has superb graphics, but also includes programming for optional sound effects. A one-player game for the PET 8K. **Order No. 0032P $7.95.**

DUNGEON OF DEATH

DUNGEON OF DEATH Battle evil demons, cast magic spells, and accumulate great wealth as you search for the Holy Grail. You'll have to descend into the Dungeon of Death and grope through the suffocating darkness. If you survive, glory and treasure are yours. For the PET 8K. **Order No. 0064P $7.95.**

ARCADE I

ARCADE I This package combines an exciting outdoor sport with one of America's most popular indoor sports:

•**Kite Fight**—It's a national sport in India. After you and a friend have spent several hours maneuvering your kites across the screen of your PET, you'll know why!

•**Pinball**—By far the finest use of the PET's exceptional graphics capabilities we've ever seen, and a heck of a lot of fun to boot.

Requires an 8K PET. **Order No. 0074P $7.95.**

CODE NAME: CIPHER

CODE NAME:CIPHER

Enjoy that same feeling of intrigue and discovery with the Code Name: Cipher package. Included are:

•**Memory Game**—Would you like to match your memory against the computer's? You can with the Memory Game.

•**Codemaster**—One player types in a word, phrase, or sentence, and the PET translates that message into a cryptogram. The other player must break the code and solve the cryptogram in the shortest time possible.

•**Deceitful Mindmaster**—This isn't your ordinary Mastermind-type game. You must guess the five letters in the hidden code word.

•**Code Breaker**—Cracking this code won't be as easy as cracking walnuts. You'll need to flex your mental muscles to win this game.

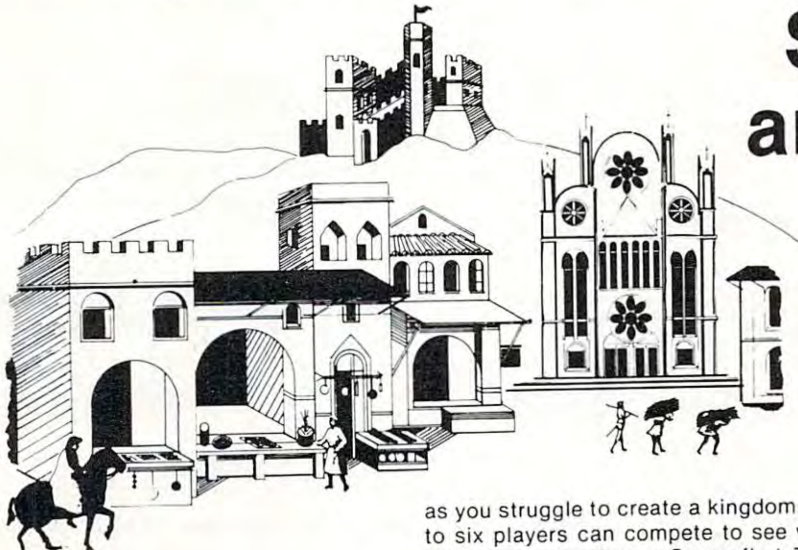
If you want a mental challenge, then Code Name: Cipher is for you. For the 8K PET. **Order No. 0112P. $7.95.**

Instant Software™

*A trademark of Commodore Business Machines

PETERBOROUGH, N.H. 03458
603-924-7296

Captivate Yourself.



Santa Paravia and Fiumaccio

The most captivating and engrossing program ever made for the PET*

SANTA PARAVIA AND FIUMACCIO Become the ruler of a medieval city-state as you struggle to create a kingdom. Up to six players can compete to see who will become the King or Queen first. This program requires a PET 16K. **Order No. 0175P.$9.95.**

It is the dawn of the 15th Century; you rule a tiny Italian city-state. Your goal: The Crown!

Up to six players can compete as rulers of neighboring cities. You control the grain harvest, feed your serfs, set tax rates, dispense justice and invest in public works.

The future of your realm will depend on your decisions. If they are wise, your city-state will grow and you will acquire loftier titles. If your rule is incompetent, your people will starve and you may be invaded by your neighbors.

How will *you* rule your kingdom? Will you be an enlightened leader—or an unscrupulous despot? Only you can answer that question—with Santa Paravia and Fiumaccio.

CHIMERA

CHIMERA If you think the legendary Chimera was hard to handle, wait until you try the Chimera package. Included are:

•**Reflex**—Round and round the little white ball rolls. Only fast reflexes can guide it into the center of the maze.
•**Dragon**—You'll have to shoot down those pesky, fire-breathing dragons with your cannon. If you succeed your castle will be safe, if not it will mean a call to your fire insurance company. For one player.
•**Dungeon**—A very punctual guard comes down to the dungeon every day to torture you. This means that you have only thirty seconds to dig your way under the castle to freedom. For one player.
•**Dragon Hunt**—You must go forth and slay a fire-breathing dragon. The only thing that will protect you from the flames is your shield, if you know when to use it. For one player.
•**Dropoff**—You must make your opponent's men "dropoff" the board by moving and firing your own men. For one or two players. **Order No. 0110P. $9.95.**

PET DEMO I

PET DEMO I You can give yourself, your family, and your friends hours of fun and excitement with this gem of a package.

•**Slot Machine**—You won't be able to resist the enticing messages from this computerized one-armed bandit.
•**Chase**—You must find the black piece as you search through the ever-changing maze.
•**Flying Pheasant**—Try to shoot the flying pheasant on the wing.
•**Sitting Ducks**—Try to get your archer to shoot as many ducks as possible for a high score.
•**Craps**—It's Snake Eyes, Little Joe, or Boxcars as you roll the dice and try to make your point.
•**Gran Prix 2001**—Drivers with experience ranging from novice to professional will enjoy this multi-leveled race game.
•**Fox and Hounds**—It's you against the computer as your four hounds try to capture the computer's fox.

For true excitement, you'll need a PET 8K. **Order No. 0035P $7.95.**

DOW JONE$

DOW JONES Up to six players can enjoy this exciting stock market game. You can buy and sell stock in response to changing market conditions. Get a taste of what playing the market is all about. Requires a PET with 8K. **Order No. 0026P $7.95.**

We Guarantee It!

Instant Software Guarantee

OUR PROGRAMS ARE GUARANTEED TO BE QUALITY PRODUCTS. IF NOT COMPLETELY SATISFIED YOU MAY RETURN THE PROGRAM WITHIN 60 DAYS. A CREDIT OR REPLACEMENT WILL BE WILLINGLY GIVEN FOR ANY REASON.

Prices subject to change without notice.

TO ORDER

SEE YOUR LOCAL INSTANT SOFTWARE DEALER

OR

call now Toll-Free 1-800-258-5473

Instant Software™

*A trademark of Commodore Business Machines

PETERBOROUGH, N.H. 03458
603-924-7296

goodies. That helped tremendously, as hand assembly is not one of my favorite activities.

Some of the code that might have looked like this:

```
CPX T       (compare T to X-reg.)
BEQ STP     (if result = 0, goto STP)
JMP PK      (if not, got PK)
STP:        ...
```

was fixed up to a reasonable

```
CPX T
BNE PK
STP:        ...
```

while avoiding branching around branches and stepping on your own feet.

```
STX KY1
JMP END  (direct or indirect)
```

was changed to;

```
STX KY1
BPL END
```

as an appropriate way to write an unconditional GOTO. I wanted to store a value in KY1 and go to END. It's unnecessary to use a JMP instruction in a program this size and use of JMP makes it harder to move the code. Indirect jump, while helping relocatability, is an overkill in a minor procedure. Again the solution is simple. If you know that the flag at that point in the program is, for instance, always positive, a simple "branch if plus" improvises the unconditional GOTO. You may want to read Butterfield's article in Nov.-Dec., 1980 **COMPUTE!** which describes addressing modes and a "reach of instructions".

These examples do not now exist in the program, as they have been superseded by better code. But they illustrate silly trouble one can get in. Needless to say, after Jim Butterfield helped me the code became clearer and shorter.

Now, there is more to programming than just uncluttered, short code, whether the code is in machine, Basic, Pascal, or any other language. And this was the most important bit of advice I got. Combine a program and its tables into a coherent structure. Its simplicity should leap at you. And simple means less work, and perhaps no backward counting in hex...

I will illustrate this concept with a simple Basic example. It shows how a little bit of code can do lots of work. First, the clumsy way. We may set up a table of 10 values from Herman's list and loop through the table until the value of the bit pattern on the user port equals one of the values in the array, in which case the loop index (J) becomes the result. Like this:

```
JS = value of 4 bits
PV( ): 3 9 11 10 13 15 14 5 7 6
FOR J = 0 TO 9
IF JS = PV(J) then K1 =J: get out
NEXT J
```

There is nothing wrong with this type of coding except that it is inefficient. The program may have to loop ten times for each joystick before finding one matching value. A much neater structure is the one used in the routine shown in this article. Its Basic equivalent is:

```
JS = value of 4 bits
KY( ): x x x 0 x 7 9 8 x 1 3 2 x 6 5
K1 = KY(JS)
```

Note the absence of the unnecessary loop. Note the reduction in code. We now have a table of sixteen values, ten of which are the numeric keypad values (0 to 9). In my program x's have been filled with 10 ($0A). Depending on the value of the four bits at the user port the program addresses that position in the table which corresponds to the port value. Thus if the four-bit value at the user port is 9 the program, by use of the Y register, picks up the 9th value in the table (counting from zero) which in this case is 1. This kind of coding is in the disassembled listing at $7023-7027 and $702E-7032.

The rest of the code has to do with housekeeping: testing the stop key at $FFE1 and shifting bits around. The value seen at the user port is saved on the stack (line $701F) so that subsequent shifting and masking will not destroy the value, as it is needed in testing both joysticks. It is brought back in $7028 and reused.

I would like to add two more ideas that will prevent you from going crazy. When you start coding leave lots of room between logical sections of your code and fill the gaps with $EA, that is, for no operation, it is a filler. (The less you know the more EAs you need. Start with fifty). As you discover mistakes you will be able to use the room while expanding code. When the program works, eliminate EAs to create a compact package of code. Try to get an assembler-disassembler. It's invaluable. It makes the work possible. You will be able to concentrate on thinking instead of hassling with hand assembly. SUPERMON, for instance, is available from Pet Program Exchange, P.O. Box 561, Montgomeryville, Pa. 18936. *(Editor's Note: Price is $1.00 for tape and $1.00 for program.)*

If your sanity is still intact after a couple dozen lines of code you'll have a good laugh as you come to grips with that very strange, but useful, language of your Pet.

Changes for old-ROM Pets:

1. In BASIC program:

replace PEEKs and POKEs to locations 52 and 53 by 134 and 135.

first DATA line has two 17-s and one 18,

Replace 17 by 8. Replace 18 by 9.

third DATA line has two 17-s. Replace by 8.

2. In the ASSEMBLY listing: replace $11 by $08 in four places and replace $12 by $09 in one place.

3. In the MONITOR listing: same thing as in the assembler.

```
B*
     PC  IRQ  SR AC XR YR SP
.;  784B 7E11 30 78 5E 34 F0

.
7000 A9 34        LDA #$34
7002 18           CLC
7003 65 11        ADC $11
7005 85 11        STA $11
7007 90 02        BCC $700B
7009 E6 12        INC $12
700B A9 00        LDA #$00
700D 8D 43 E8     STA $E843
7010 20 E1 FF     JSR $FFE1
7013 AD 4F E8     LDA $E84F
7016 AE 37 03     LDX $0337
7019 F0 04        BEQ $701F
701B C9 FF        CMP #$FF
701D F0 F1        BEQ $7010
701F 48           PHA
7020 29 0F        AND #$0F
7022 A8           TAY
7023 B1 11        LDA ($11),Y
7025 8D 38 03     STA $0338
7028 68           PLA
7029 4A           LSR
702A 4A           LSR
702B 4A           LSR
702C 4A           LSR
702D A8           TAY
702E B1 11        LDA ($11),Y
7030 8D 39 03     STA $0339
7033 60           RTS
.?
.
.:  7034 0A 0A 0A 00 0A 07 09 08
.:  703C 0A 01 03 02 0A 04 06 05
.?
```

Figure 1

```
.
.:  7000 A9 34 18 65 11 85 11 90
.:  7008 02 E6 12 A9 00 8D 43 E8
.:  7010 20 E1 FF AD 4F E8 AE 37
.:  7018 03 F0 04 C9 FF F0 F1 48
.:  7020 29 0F A8 B1 11 8D 38 03
.:  7028 68 4A 4A 4A 4A A8 B1 11
.:  7030 8D 39 03 60 0A 0A 0A 00
.:  7038 0A 07 09 08 0A 01 03 02
.:  7040 0A 04 06 05 20 20 20 20
```

Figure 2

I'd like to thank Jim Butterfield for helping me take the plunge into the machine code…for the SUPERMON…lots of patience…and an incredible ability to share his knowledge.

```
100 REM------------------------------
110 REM SUBROUTINE FOR TWO JOYSTICKS
120 REM ATTACHED AS IN H.HERMAN'S
130 REM ARTICLE - COMPUTE V.1,#4,P.89
140 REM
150 REM       BY ELIZABETH DEAL
160 REM
170 REM FOR LOADING MACHINE CODE INTO
180 REM   SECOND TAPE BUFFER REMOVE
190 REM   'REM' FROM LINE 250
200 REM TO PUT CODE AT TOP OF PET
210 REM   USE LINES 260-300
220 REM SEE ARTICLE FOR CHANGES TO
230 REM   OLD ROM PETS
240 REM------------------------------
250 :REM SP=67:AD=826:FORI=ADTOAD+SP:
     ¬READV:POKEI,V:NEXT:GOTO500:
     ¬ <TO POKE823,0
260 IFPEEK(894)=1GOTO300
270 SP=67:AD=PEEK(52)+256*PEEK(53)-SP
280 FORI=ADTOAD+SP:READV:POKEI,V:NEXT
290 AH%=AD/256:AL=AD-256*AH%:POKE52,AL:
     ¬POKE53,AH%:POKE894,1:CLR
300 AD=PEEK(52)+256*PEEK(53)
310 :
320 DATA 169,52,24,101,17,133,17,144,2,
     ¬230,18,169,0,141.67,232
330 DATA 32,225,255,173,79,232,174,55,3,
     ¬240,4,201,255,240,241,72
340 DATA 41,15,168,177,17,141,56,3,104,
     ¬74,74,74,74,168,177,17
350 DATA 141,57,3,96,10,10,10,0,10,7,9,
     ¬8,10,1,3,2,10,4,6,5
360 :
370 REM------------------------------
380 REM POKE823,W
390 REM   W=0  NO WAIT FOR JS MOTION
400 REM   W<>0 WAIT UNTIL MOVED
410 REM GET KEYPAD EQUIVALENTS OF JOY-
420 REM STICK MOTION FROM 824 AND 825
430 :
440 REM 0=BUTTON    1=LEFT-DN  2=DN
450 REM 3=RIGHT-DN  4=LEFT     5=NONE
460 REM 6=RIGHT     7=LEFT-UP  8=UP
470 REM 9=RIGHT-UP
480 REM------------------------------
490 :
500 POKE823,0
510 SYS(AD):PRINTPEEK(824),PEEK(825):
     ¬GOTO510
```

©

Machine Language: The Wonderful Wedge

Jim Butterfield

Adding new commands to Basic seems an impossible task at first glance. The Basic interpreter is frozen forever in ROM chips, and unless you're the adventurous type who can program your own EPROM chips, it seems that there's no way in.

It can be done. A small but important part of the Basic interpreter is located in RAM memory. It's written there during system initialization and is available for you to change.

The subroutine is called **CHRGET** (Character Get), and all 6502 Microsoft Basic implementations use it. Every time the Basic interpreter wants to get a character from the Basic statement it is executing, it calls CHRGET.

Here's where you can find subroutine CHRGET in some 6502 systems:

KIM -	C0 to D7 hexadecimal
SYM -	CC to E3
AIM -	BF to D6
OSI -	BC to D3
Apple -	B1 to C8
Early PET -	C2 to D9
PET/CBM -	70 to 87

Our description here will refer to the PET/CBM version, Upgrade and subsequent ROMs.

How it works

Let's look at the CHRGET subroutine in detail.

```
0070 E6 77  CHRGET  INC POINTER
0072 D0 02          BNE CHRGOT      ;skip next instruction
0074 E6 78          INC POINTER + 1
```

Locations 77 and 78 contain the address of the last Basic character obtained. The above coding bumps the pointer to the next address, adjusting the high order address if necessary.

```
0076 AD xx xx CHRGOT LDA xxxx       variable address
```

The address indicated with xxxx above normally points at your Basic program or at a direct Basic statement you have typed in. Note that the address itself has been modified by CHRGET, above.

```
0079 C9 3A          CMP #':         ;ascii colon or higher?
007B B0 0A          BCS EXIT        ;yes, exit subroutine
```

The above coding tests two things. If the new character is a colon, meaning end of Basic statement, we will exit with the Z flag set to one. If the new character is higher than ASCII 9 (hex 39), we will exit with the Carry flag set to one. The meaning of these flags will be discussed in a moment.

```
007D C9 20          CMP #'          ;is it a space?
007F F0 EF          BEQ CHRGET
```

We know that Basic ignores spaces; this is where it happens. If we find a space, we go back and get another character.

```
0081 38             SEC
0082 E9 30          SBC #$30
0084 38             SEC
0085 E9 D0          SBC #$D0
```

This seems to be a curious bit of coding: we subtract 256 from the A register, in two steps, which leaves it with its original value! The point is this: if the A register contains a value less than ASCII zero (30 hex), the Carry flag will be set to one; otherwise, it will be cleared to zero. The Z flag, too, will be affected: it will be set if we have obtained a binary zero.

```
0087 60             RTS
```

What the flags mean

The flags are often checked by the calling routines. The Z flag will be set on if we have found an ASCII colon (end of statement) or a binary zero (end of Basic line).

The Carry flag will be cleared to off if the character is an ASCII numeric, zero to nine (30 to 39 hex); otherwise it will be set on.

How the subroutine is called

CHRGET is called many times during the interpretation of a Basic program or a direct statement. It normally obtains data from the active program; but it is also used to obtain information from DATA statements or keyboard input during READ or INPUT activities. In such cases, the pointer at 77 and 78 is swapped out temporarily.

The Basic interpreter also frequently calls CHRGOT (address 0076) to re-obtain and check a previously obtained character.

From time to time, the pointer at 77 and 78 is used as an indirect address by the interpreter; when we start tampering with the coding of the subroutine, we must be sure to leave the pointer intact in its normal place.

Finally, there is a rare call that is made to the subroutine at address 7D (Compare to space); it doesn't happen often, but we must watch for it.

Keep in mind that the subroutine does not affect the X or Y registers.

Wedging it in

To fit in the extra features, we must "patch" the CHRGET program and connect it to our own code. The patch will destroy some of the existing code, of course, and we must carefully replace it.

There are two places we can insert the patch: at the beginning of CHRGET, or a little distance past CHRGOT. The first location will go into action only when a new character is called up by the interpreter. The second location would be invoked more often, since CHRGOT is called to recheck a previously obtained character.

Let's use the first location; and let's put in a simple do-nothing wedge for starters. Call up the Machine Language Monitor and set up the following memory locations as shown:

```
027A:   E6 77 D0 02 E6 78 4C
0282:   76 00 xx xx xx xx xx
```

The first six locations exactly match the coding at CHRGET. Now we'll put in the patch with:

```
0070:   4C 7A 02 02 E6 78 AD xx
```

Leave the Machine Language Monitor and play with Basic for a moment. Everything still works. It looks like we have found a way to penetrate Basic... but we haven't done anything yet.

A tiny example

Let's write a very small wedge to recognize an "@" sign and break to the monitor if it is seen. Not much in the way of power, but it will show how the technique is used. We'll continue to use the patch at 0070.

To get the character we plan to analyze, we'll have to use indirect, indexed addressing. The pointer is of course at 77 and 78, and we must set the Y register to zero. Since we must not affect the Y register, we must first save its contents, and restore them before we finish.

So our coding will follow the following pattern: STY WORK, to save Y; LDY #0, LDA (POINTER), Y to get the character; LDY WORK, to restore Y; CMP #'@, to check for the @ character in A; BEQ BREAK if we find it; and JMP CHRGOT to return if not. BREAK will have the BRK instruction to go to the Monitor. Let's do it.

```
027A:   E6 77 D0 02 E6 78 8C A0
0282:   02 A0 00 B1 77 AC A0 02
028A:   C9 40 F0 03 4C 76 00 00
```

We have arbitrarily picked address 02A0 as our Y Save location. Now the patch to implement the wedge:

```
0070:   4C 7A 02 xx xx xx AD xx
```

Return to Basic. Try statements which do not contain the @ sign, and others which do.

Final remarks

You're ready to try your hand at more ambitious wedge inserts. Be careful: remember to save X and Y if you use them, and restore them later. Keep in mind that the larger your wedge program, the slower Basic will run. Look for quick tests: for example, many wedge programs will exit instantly unless the statement was input as a Direct command... this can save a lot of time on a running program.

Watch that you don't conflict with other wedge programs, like Trace, Toolkit, or the DOS wedge program. It takes a lot of careful coding; but the results can be dramatic. ©

The Single-Board 6502

Eric Rehnke

The 6500 Family—Just where is it going?

Six years ago a star was born. The 6500 family was the product of a chasm between some Motorola design engineers and their management. They broke away and joined MOS Technology, a calculator manufacturer, to help produce one of the most revolutionary pieces of silicon to ever roll off a drawing board. The main idea was to simplify a chip design as much as possible by removing everything but the bare necessities. This led to a chip which was much smaller than its contemporaries and was in turn much easier to produce. Since chip yields were high, the end price could be much lower than the competition. The 6502 was going for $25 when the 6800 had to sell for around ten times that much. Combine that low price with an elegant architecture and a chip which was both easy to learn and use and what did you get? A sure winner, that's what.

And if all that weren't enough, the factory was selling direct to individuals in single quantities. This alone was completely unheard of at the time. But, since MOS Technology management had to play a game of catch up to assure themselves of an adequate number of new design ins, it was a shrewd and well calculated move. It sealed the fate of the 6500 family. It would succeed. But MOS Technology was determined not to give their competition any breathing room.

At a time when most of the industry were selling evaluation kits to help folks become familiarized with their devices, MOS Technology brought out the KIM-1, a system with features that were unheard of. A scanning hex keyboard and display, a built in audio-cassette and terminal interface. All for $245...and it was assembled and burned in at the factory to boot! The marketplace LOVED it! Now we had an easy way to evaluate the 6502 without going through the system design phases on our own.

As an aside, shortly before the KIM-1 came out, I had designed a system which would let me evaluate the 6502 chip. I designed a hex keyboard and display that would allow me to enter and examine data in memory as well as start a program running. Since I knew nothing about software at the time, the system was designed using TTL gates and counters, stuff I was familiar with at that time. My design would have taken *AT LEAST 50 TTL chips* to implement. And there were no means of saving data to an audio cassette or communicating with a terminal. Well, you can imagine how I felt when I first saw a KIM-1 with all the stuff I wanted plus features I didn't even realize I needed. All assembled, tested and at a price I couldn't pass up. As I look back, I can't remember ever making a better investment.

The design cleverness of the KIM-1 only became apparent when users started doing the most amazing things with the system. The most memorable early contributions were those of Jim Butterfield from Toronto, Canada. I can still remember my fascination with his realtime moonlander program. I just couldn't believe the KIM-1 was performing these feats of "magic".

From the early days it became increasingly apparent that 6502 users were the elite of the hobby world. No other computer users were so creative and so willing to share their discoveries. Although Jim Butterfield deserves a lion's share of the credit for helping the KIM-1 become so popular, it was Rick Simpson, then Product Manager of the KIM-1 line, who really got things rolling by getting a few of the early KIM-1 users together. Rick thought up the idea for a KIM-1 user group newsletter and helped me get the original KIM-1 user notes started.

The 6500 family was proving very popular in real time systems such as video graphics machines because of its high throughput capability. It was one of the faster 8-bit chips and also very cost effective to design in because of the low number of support chips that were necesary to get a system together. This made it attractive for small dedicated systems use. Its elegant architecture and instruction set made it a programmer's dream. Since the influence of the DEC PDP-11 was so obvious, even down to some of the assembler mnemonics, people who were familiar with that popular mini could easily migrate downward to an 8-bit system and not have to start from scratch on the learning curve.

A short while after the 6500 family was introduced, thought was given to what should follow. Design specification work began on what was to be called the 6516 Psuedo-16 bit cpu. This processor had a rather unique design feature that gave the chip a dual personality! A single bit in the status register could be set to change the cpu from one that looked exactly like the 6502 to a psuedo-16 bit device with advanced instructions and even greater capability than the 6502.

But, it just wasn't in the cards for this advanced chip to ever become a reality. There was some talk of Synertek doing the chip, but, that never panned out either.

At this point in time, I don't see any real possibility of there being an expanded psuedo (or real) 16 bit cpu based on the 6502. Two years ago, such a chip would have been a hit. Today, it would be a disaster. Sorry bunky.

However, the 6500 family hasn't stopped growing. It just isn't growing in the same direction as some of the other semiconductor manufacturers' products. A lot of the development of the 6500 family of devices is being made in the area of single-chip microcomputer devices. These "computers-on-a-chip" will be increasingly important in the 80's as peripheral processors for the larger 16 bit cpu's. A system with 3 or 4 of the 16 bitters could conceivably have up to 30 or 40 of the single-chippers handling the more mundane system functions.

Admittedly, the thought of single-chippers probably isn't as glamorous as some of the new 16-bitters (such as the 68000), but, if you have a practical use for small dedicated systems around your home or lab, these small "all-in-one" devices will really fill the bill. Rockwell is planning to come out with single-chip devices that can talk to off chip EPROMS so you can easily develop your own programs and not have to buy a thousand of the little buggers just to automate your home energy management system, for example. And since the RAM, and I/O are on one chip (along with the cpu), the system design turns out to be a lot simpler.

And I've just heard a rumor that MOS Technology has announced a new chip called the 6508. Supposedly it has some on-board RAM and one 8-bit I/O port. But, I won't know for sure until the data sheet arrives. Now, if the full 16-bit address bus is present on the 6508, its eight bit I/O port could be used as a bank select for those applications requiring more memory than the normal 64K. Of course, it would be most useful at the lower end of the applications spectrum where the 6508 and an EPROM could form a very low-cost system.

So don't despair, all's not lost. The 6500 will live on.

6502 Word Processor (?)

For quite some time I have wanted a word processor for my HDE expanded KIM system. Now the text editor that comes with the Hudson Digital Electronics disk system is probably the best there is for editing assembler source files. But, it has some limitations which make it a bit awkward when handling articles or doing letter writing. Fortunately, another HDE system user, Chuck Kingston, also felt the need to have a processing capability and he went ahead and wrote one.

His word processor is actually in two parts — the input editor and the output formatter. The editor uses line numbers (which makes the files readable with the HDE editor) but the lines in the editor file don't necessarily correspond to the output of the text formatter. The column width set command (which can either be imbedded in the text file or set up upon entry to the text formatter) determines the printing buffer length. Words are pulled from the input file and stuffed into this printing buffer until it's filed or a new line command has been encountered in which cases, the printing buffer will be dumped to the printer. So the line numbers are essentially ignored when the text is output to the printer.

The editor includes commands for auto line numbering, block moves, string searching and replacing, tabbing and so forth. The text formatter allows margins and column size to be set, files to be chained in from disk, page numbers to be added, and includes lots of other features too numerous to mention.

I think that the main feature of this system which makes it more than just another text editor is the way multiple lines can be inserted anywhere in the file, even between words on the same line. Normal text editors will not let you insert more text on one line than the difference between the amount of text already contained on that line and the length of the line buffer. To insert any more you'd have to delete some of the text already there. Chuck Kingston's word processing system automatically saves any text overrun to new lines and lets you insert text to your heart's content — up to the limits of memory, of course.

This software has added a new dimension of usability to my computer and I can see it being used a lot in my line of work. Now I'm going to have to figure a way to get my system out to the patio so I can work in the sunshine.

If you've been using the HDE text editor, you'll have no problem learning this word processor. I haven't tried to teach my wife how to use this software yet so I can't say how long it would take to teach someone who is not a constant computer user.

At this time, the system is available for KIM systems with the 5'' HDE disk system. The price of $75 (NY residents must add appropriate tax) includes the program object code AND the manual on disk. That's right, the manual is included on the disk as a number of text files. Besides being less expensive to produce, it has the advantage of providing examples of about every command in the system for your inspection. Versions for the 8'' HDE (KIM and SYM) and 5'' HDE SYM systems should be available for the second quarter of 1981. AIM disk versions and cassette versions for the KIM, SYM, and AIM machines are planned for sometime in 1981.

If you do any writing at all, even just personal letters, you'll want a copy of this text editing software for your HDE disk system.

The text editing system is available from Charles Kingston, 6 Surrey Close, White Plains, NY 10607.

The reason for the question mark in the title of this article? Well, when I started to write it, I had wondered about what the difference was between a word processor and a text editor. I tried to figure out in my mind just what this particular system was and what it could be honestly called. After about two days of thought on the subject, I've decided that I don't really know the difference and it may not really matter anyway since everyone will have their own opinion. Since I can manipulate and insert text fairly comfortable with this particular editing software, I think it can be referred to as a word processor. If it were clumsy to operate I might call it a lousy text editor. So you decide...

WHAT IS THE DIFFERENCE BETWEEN A WORD PROCESSOR AND A TEXT EDITOR?????????

A Low Cost Controller Development System

The AIM 65, with an assembler ROM, makes a very usable and low-cost development system, as long as whatever is being developed is meant to reside on the AIM. However, if you should want to develop a dedicated controller system, the AIM doesn't really have the proper tools. What's needed is some way to communicate with the controller board to somehow download the object code into it for testing. But, up until now, there have been no low-cost add-on tools to enable the AIM to handle this kind of duty.

Well, luckily for those of us who need such tools, that is no longer the case. A company called R.J. Brachman Associates Inc. has come up with not only the controller development tools necessary but the controller board as well.

Let's talk about the controller first. Basically, it's a 4.5" x 6" p.c. board that contains a 6503 microprocessor, a 1MHZ xtal, two 6522 VIA chips, two 2114 RAMs, and a socket for a 2716 (2Kx8) EPROM. There's even a power supply on-board that will accept 9 to 18 volts of A.C. The I/O comes off the board from a 44-pin edge connector. As you know, the 6503 has a total addressing capability of 4K bytes — more than enough for a small controller system. The board layout is very clean and professional looking and a solder mask is included on both sides of the board. There's even an LED on the board to indicate power-on.

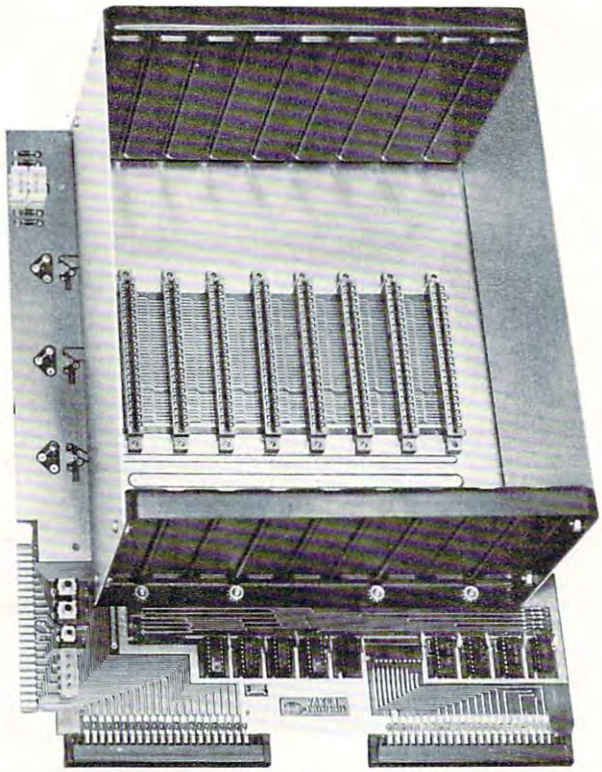
As part of the development package, they have an In Circuit Emulator that takes the place of the 6503 CPU in the controller and maps the lower 2K of the controller (everything but the EPROM) into the address space of the AIM. This means that you can directly access everything in the controller's memory space from the AIM. Think about that!

The controller hardware and software can be fully developed and debugged using the tools already resident in the AIM (like the assembler, the software trace and breakpoint routines). No need to download to another system and work in the blind anymore.

And when the system is developed, R.J. Brachman also has an EPROM programmer which plugs onto the controller and lets you make things more permanent. This is a very nicely integrated development system at a surprisingly low price. The In Circuit Emulator sells for $95, the EPROM programmer attachment sells for $45 and the controller board sells for from $25 for a bare pc board w/manual to $149 fully assembled with the on-board power supply. Pennsylvania residents need to add 6% tax.

The key to the low cost and high usability of this developmment system is in the use of the 6503 as the controller CPU. Because it can only address 4K of memory, all of it can easily be accessed by the host

VAK-1 MOTHERBOARD



We also carry:

SYM-1	**$229.00**
AIM-65 w/1K	**389.00**
AIM-65 w/4K	**439.00**

We also do custom hardware and software for the 6502 microprocessor

PRICE: **$139.00**
Call or write for shipping charges and our complete catalog.

The VAK-1 was specifically designed for use with the KIM-1, SYM-1 and the AIM 65 Microcomputer Systems. The VAK-1 uses the KIM-4* Bus Structure, because it is the only popular Multi-Sourced bus whose expansion boards were designed specifically for the 6502 Microprocessor.

SPECIFICATIONS:

- Complete with rigid CARD-CAGE
- Assembled (except for card-cage). Burned in and tested.
- All IC's are in sockets
- Fully buffered address and data bus
- Uses the KIM-4* Bus (both electrical Pin-out and card size) for expansion board slots
- Provides 8 slots for expansion boards on 1″ centers to allow for wire-wrap boards
- Designed for use with a Regulated Power Supply (such as our VAK-EPS) but has provisions for adding regulators for use with an unregulated power supply.
- Provides separate jacks for one audio-cassette, TTY and Power Supply.
- Board size: 14.5 in. Long x 11.5 in. Wide x 8 in. High
- Power requirements; 5V.DC @ 0.2 Amps.

**KIM-4 is a product of MOS Technology/C.B.M.*

RNB ENTERPRISES INCORPORATED

2951 W. Fairmount Avenue
Phoenix, AZ 85017
(602) 265-7564

master charge VISA

computer through the In Circuit Emulator. Another benefit is that since the 6503 only uses a 12 bit address bus, and the top 4 bits are ignored, programs which reside in *ANY* 4K block can be installed in a 2716 and run in the controller without needing to change any addresses at all!

If you've ever tried to develop a small controller without tools like these, you'll know why I'm so enthusiastic about them. Get more information from R.J. Brachman Assoc. Inc., POB 1077, Havertown, PA 19083 (phone 215-622-5495).

Pentominoes— The Ultimate Demo

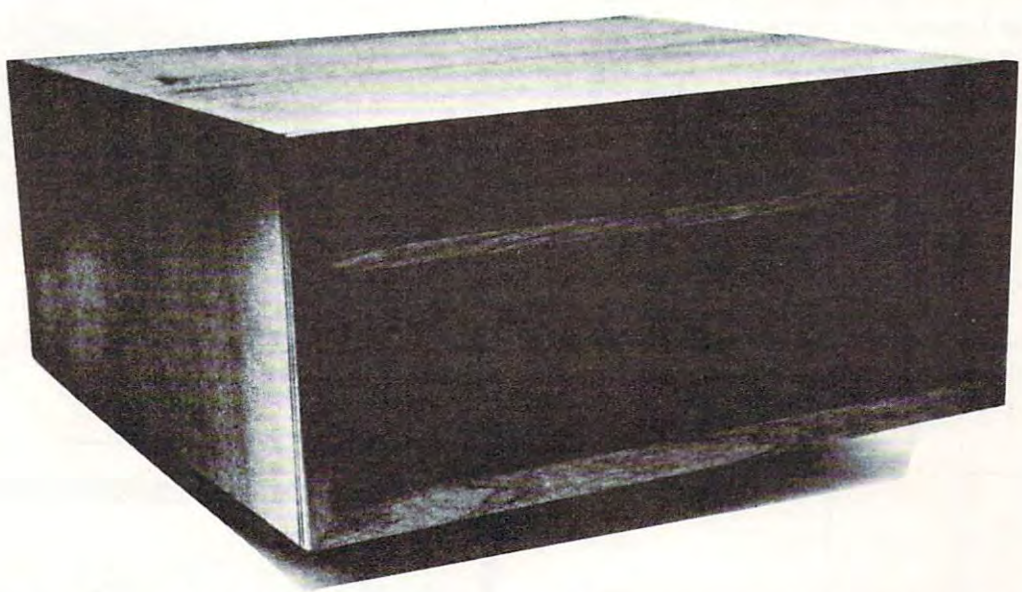
Every so often a non-computer type comes over to my house and our conversation may casually turn to the subject of computers. If I mention that I have a computer, he'll eventually want to see it in action. Unfortunately, watching an assembler or an editor doing its thing seldom excites the unknowing onlooker. As a result, I have been searching for the ultimate demonstration program. And...I think I just may have found it.

The "ultimate demo" should not require any input from the onlooker and needs to show how powerful the computer is in solving problems.

Pentominoes, a program from the 6502 Program Exchange meets these specifications very nicely. **Pentominoes** is the computeriztion of a jigsaw puzzle that has fascinated and frustrated math and logic types for years. The puzzle consists of 12 pieces that must be fitted together to form a rectangle. Each of the pieces represents one of the twelve possible patterns that can be assembled with five unit squares.

For example, here is one possible combination. There are eleven others.

Since each pattern consists of 5 units and there are 12 such possible patterns, the total area of the rectangle is 60 units. A 6 by 10 arrangement has been popular with puzzle solvers, but, other arrangements (5 by 12, 4 by 15, and 3 by 20) are also possible. The 3 by 20 seems to be the most difficult of the possible rectangle arrangements. According to the

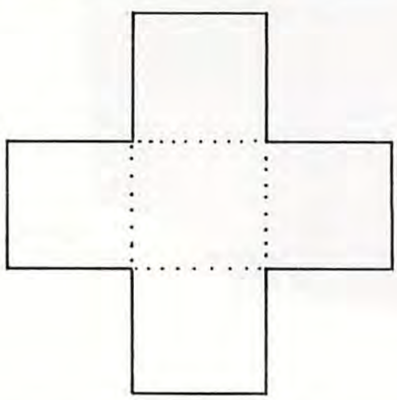


program documentation, out of 1,000,000,000,000,000 possible placement combinations of the 3 by 20 rectangle, only 2 solutions exist.

Pentominoes is a 6502 machine language program that solves the jigsaw puzzle by trying all possible combinations of the pieces for a given rectangle arrangement until it finds a combination that fits together. The desired rectangle size can be chosen by the user as well as whether all, some, or none of the attempts are displayed on the users Ascii output device. Modification information, as well as a complete source listing, is provided to enable you to configure Pentominoes to just about any 6502 system/terminal combination (including memory mapped video).

To let Pentominoes be displayed on any Ascii output device, each pattern is represented by a particular combination of letters arranged in the same pattern.

For example, the cross pattern



is represented by the letter 'A' in the same pattern.

```
 A
AAA
 A
```

On my Hazeltine 1500 terminal, Pentominoes clears the screen and homes up the cursor every time it prints an attempt. Since the terminal runs at 9600 baud with KIM, and the attempts are happening at about 2 or 3 times per second, it looks like an animation and is quite spectacular to view.

Pentominoes costs $15 and is available for KIM, SYM, AIM, AND TIM systems at memory locations $2380-$2801 or $0380-$08A1. Or send $1 for a complete catalog to:

6502 Program Exchange
2920 Moana Ln.
Reno, Nevada 89509

©

KIM Tidbits

Expanding Your System

Harvey B. Herman
Department of Chemistry
University of North Carolina
at Greensboro
Greensboro, North Carolina 27412

The January 1981 issue of **COMPUTE!** carried a brief review by me, of a disk operating system for an expanded KIM-1. I have been asked (can you be locked up for talking to yourself?) to supply details on how to expand KIM into a complete system including a disk drive. Even if you have reservations about this, as I am sure some readers do, read on and see how I did it. I am quite pleased with the finished project and feel it may make sense for other KIM owners to follow my lead.

First Steps

My work with microcomputers began in the summer of 1976 when I purchased a KIM-1, a small power supply and teletype (KSR-33). My justification was to have these serve as both an educational tool for me and my students and as an instrument controller. Our first project, a success, was to develop a machine language program, hand assembled by Jeffrey Schmoyer, to control a syringe pump. Hand assembly is a drag and the next natural step was addition of extra memory in order to run an assembler. An article in 6502 Users Notes (November 1976) encouraged me to order a 4K $100 memory board (S.D. Systems, Dallas, TX 75228) and attach it directly to the KIM bus.

At this point I was able to run a small assembler and tiny BASIC. I quickly saw the value of hybrid programs which mixed experimental control (in machine language) and data analysis (in BASIC). However, in general, the later required a floating point BASIC. Microsoft 8K BASIC was, in late 1977, just being offered by Johnson Computer (Medina, OH 44256). This program needed additional memory and I felt compelled to take the next step and add a mother board, KIMS I (Forethought Products, Eugene, OR 97402), which could support $100 8K static memory add-ons, e.g., Econoram II (Godbout Electronics, Oakland Airport, CA 94614).

Willi's Advice

Each step in the expansion of KIM this far was relatively painless with no problems and did not appear overly expensive. Now I was able to run a very nice enhanced BASIC (see previous KIM Tidbits) and a more comprehensive assembler.

The only frustration was loading from cassette tape (even at 6x normal speed using Butterfield hypertape). Obviously a conversion to floppy disk was the way to go but none of the commercial systems was in my budget range. I did have an 8" disk drive and homemade power supply left over from an aborted project. Willi Kusche (Wilserv Industries, Haddonfield, NJ 08033) noted my dilemma and constraints on a visit here in the later part of 1979. He informed me that with the addition of a disk controller, cable, and software, which he sold, I could get a disk operating system (DOS) up on my KIM. This idea appealed to me and even though the going was not perfectly smooth I'm glad I followed his suggestion and began to put the system together.

Adding The Disk

The first step was to construct the disk controller board (Versafloppy I, S.D. Systems). Table I shows the jumpers we made on the board. The kit was not too difficult to make but one chip was bad and I clumsily blew out several more before I got the board working. After that auspicious start we (Leon Stokes is we, I is me) put together a cable from the controller to the drive. The disk drive was old enough to have a non-standard connector and we could not purchase a ready-made cable. Table II shows the interconnections that were necessary, between the controller and the drive. (Innovex 220M, now Innotronics, Lincoln, MA 01773). Sad to tell on power up the head loaded but nothing else happened. We subsequently found that the original lubricant (not used now) had congealed because the drive had not been used for four years. Our man in the machine shop (Henry Teague) freed things up for us and now everything worked. I was able to load the disk Willi had sent and made a back-up copy with his utility program.

Observations

My students and I have been using this system for slightly over a year. We originally loaded a short disk boot program from tape but are now using a 16K EPROM board (Digital Research Computer, Garland, TX 75040) with just one 2708 programmed by Willi for us. We make extensive use of disk data files which are then used as input to separate data analysis programs. A simple editor program (written by Becky Efird) prepares the data for storage on disk. A listing of the very similar PET version of the program is shown as illustration of the features of the DOS. The KIM version of this program uses the USR function for monitor calls as opposed to SYS for the PET. Otherwise the programs are almost identical.

Disk data files have the decided advantage of allowing recalculation by alternate models without rekeying of data. About 128K bytes of data can be stored on each 8'' disk. While some of the space is taken up by the disk directory and DOS the latter can be optionally omitted on initial formatting. Backup of data files is done with a copy utility and a second diskette or could even be done on tape. As further insurance I am considering a second system to protect against hardware failure.

Conclusions

Would I recommend this approach to others? Yes and no. The additional cost to me for disk operation was nominal as I already had the mother board, extra memory, disk drive and power supply. However, If I was starting from scratch, a complete system based on an Apple or PET would probably have been cheaper. Furthermore, there is a potential noise problem with an exposed single board computer which other commercial expansions might minimize. I have had no problems with this but it should be mentioned.

Working within a limited budget where only a few things could be purchased at a time, I was able to assemble a very nice system. I did have some difficulties but was able to overcome. Get a commercial system if you have the money. If not, consider occasional add-ons with the object of reaching a disk based system at some future point. It really is great bringing up BASIC in less than 4 seconds. You will never be satisfied with less.

Table I

Versafloppy Jumpers for 8'' Innovex 220M Drive	Technical Function
E1 - E2	Port E3/3 to DRVSEL 4
E4 - E5	Port E3/2 to DRVSEL 3
E7 - E8	HLD to HEAD LOAD
E10 - E11	Port E3/3 to DRVSEL 4
E15 - E17	HLT(1771) to Q̄(U15)
E21 - E22	4MHz clock (8'' drive)
E24 - E25	pull up on pin 2 of U2
E32 - E33	Ports E3 to E7 (locations F0E3 to F0E7 on KIMSI)

Note -
Some jumpers are etched on boards (rev A) and may need to be cut. Most jumpers are same as given for Altair mainframe with Shugart SA800 disk drive.

Table II

Controller Versafloppy I	8'' Disc Drive Innovex 220 M	Function
36	L6	step
34	L15	direction
26	L13	device select
20	L5	index
42	L12	track zero
6	L21	write current select
18	L18	head load
40	L7	write gate
22	L8	ready
46	L17	separated data
38	L10	write data
44	L16	write protect

NOTE: Power - +5, −5, +24V, and AC made to drive.
Hard sector option on drive was disabled.
Unseparated data sent to controller on separated data line.

©

PRINTING A SYMBOL TABLE FOR THE AIM-65 ASSEMBLER

Richard F. Olivo
Biological Sciences, Smith College
Northampton, MA 01063

The assembler for Rockwell's AIM 65 makes assembly-language programming very convenient, particularly in conjunction with the excellent editor that is part of AIM 65's monitor. However, the assembler does not include an option to print the symbol table, although it does create such a table in memory. The following program is one way of decoding and printing the symbol table. In revising a program, a print-out of the symbol table can be very helpful.

On entering the AIM 65 assembler from the monitor, you are asked for the addresses that start and end the symbol table. The assembler places your answers in zero-page addresses 3A, 3B ("FROM") and 3E, 3F ("TO"). After assembly, the total number of symbols is available in addresses 0B, 0C (in high, low order). The symbol table itself consists of sequential eight-byte entries. The first six bytes of each entry are the symbol name, in ASCII characters (the assembler enters spaces if the symbol is less than six characters), and the last two bytes are the symbol's address, in hex notation.

The program to print the table reads through the table using indirect addressing indexed by Y. It establishes the variable ADDR (at locations 00 and 01), which provides the address of the first character of the current symbol. ADDR is initially set equal to the address in "FROM (3A, 3B); it is incremented by eight after each symbol is printed. For each symbol, the Y register is incremented from zero to seven to access the successive bytes of that symbol.

A second variable, COUNT (addresses 02 and 03), keeps track of the number of symbols that remained to be printed. COUNT is initially set equal to one less than the total number of symbols (from addresses 0B and 0C), and it is decremented by one after each symbol is printed. After COUNT reaches zero (the last symbol is numbered zero, which is why the initial count is one less than the total), the program exits and prints the total number of symbols in hex notation. The program uses AIM monitor subroutines to print the ASCII and hex characters. It also turns the AIM printer on and off at the start and end of the table, which I find very handy.

The listing given below places the program at locations 0200-027D, which are available on every AIM 65. The program could of course be placed in other memory locations, and it would be very convenient in a PROM. At the end of the listing, the program was run to list its own symbol table.

```
---------------------
==0000 BLANK=$E83E

==0000 CRLOW=$EA13

==0000 EQUAL=$E7D8

==0000 PRIASC=$E97A

==0000 PRIFLG=$A411

==0000 PRIHX2=$EA46

==0000 ADDR=0
==0000 COUNT=ADDR+2
==0000
       *=$0200
---------------------
INITL ADDR,COUNT,Y

==0200
;"FROM" = 3A,3B
==0200 SYMTBL
A53A   LDA $3A
8500   STA ADDR
A53B   LDA $3B
8501   STA ADDR+1
;ADDR ACCESSES TABLE

A50B   LDA $0B
8502   STA COUNT
A50C   LDA $0C
8503   STA COUNT+1
==0210
;COUNT=SYMBOLS TO GO
C603   DEC COUNT+1
;FIRST SYMB=0, NOT 1

A000   LDY #0
;INDX 8 BYTES/SYMBOL
A980   LDA #$80
8D11A4 STA PRIFLG
;TURN PRINTER ON
2013EA JSR CRLOW
2013EA JSR CRLOW
;SKIP 2 LINES AT TOP
---------------------
MAIN LOOP

==021F SYMLP
B100   LDA (ADDR),Y
C006   CPY #6
;BYTES 0-5 =ASCII
F007   BEQ SPACE
;PRINT 6 ASCII CHAR.
207AE9 JSR PRIASC
C8     INY
4C1F02 JMP SYMLP

;PRINT SPACE & EQUAL
==022C SPACE
48     PHA
203EE8 JSR BLANK
20D8E7 JSR EQUAL
203EE8 JSR BLANK
;NEXT 2 BYTES = HEX
68     PLA
2046EA JSR PRIHX2
C8     INY
B100   LDA (ADDR),Y
==023D
2046EA JSR PRIHX2
2013EA JSR CRLOW
;HAVE PRINTED 1 LINE
---------------------
DECR COUNT & TEST

C603   DEC COUNT+1
A9FF   LDA #$FF
C503   CMP COUNT+1
;FF = BORROW
D006   BNE NXTADR
C602   DEC COUNT
==024D
C502   CMP COUNT
;FF = DONE
F012   BEQ DONE
---------------------
UPDATE ADDRESS

==0251 NXTADR
18     CLC
A500   LDA ADDR
;LOW BYTE
6908   ADC #8
8500   STA ADDR
A501   LDA ADDR+1
;HIGH BYTE
6900   ADC #0
8501   STA ADDR+1
A000   LDY #0
4C1F02 JMP SYMLP
---------------------
PRINT TOTAL & EXIT

==0263 DONE
2013EA JSR CRLOW
A50B   LDA $0B
2046EA JSR PRIHX2
A50C   LDA $0C
2046EA JSR PRIHX2
;PRINT TOTAL,SKIP LN
2013EA JSR CRLOW
==0273
2013EA JSR CRLOW
A900   LDA #0
8D11A4 STA PRIFLG
;TURN PRINTER OFF
4C82E1 JMP $E182
;JUMP TO MONITOR

         END

BLANK  = E83E
CRLOW  = EA13
EQUAL  = E7D8
PRIASC = E97A
PRIFLG = A411
PRIHX2 = EA46
ADDR   = 0000
COUNT  = 0002
SYMTBL = 0200
SYMLP  = 021F
SPACE  = 022C
NXTADR = 0251
DONE   = 0263

000D
```

©

New Products

WordPro Plus Packages For Commodore Computers

Professional Software Inc. of Needham, Mass. has announced the availability of WordPro 3 Plus and WordPro 4 Plus word processing software for use on Commodore Business Machine computers.

In addition to all of the highly advanced features currently found in WordPro 3 and WordPro 4, many new capabilities are now easily accessible. New features include: Math Functions (adding and subtracting columns of numbers within text), Superscripts and Subscripts, Bold Overstrike, Exit to Basic, Variable Lines per Inch, Additional Pitch Settings, Audible Feedback, Pause Command, and Simultaneous Input/Output (WordPro 4 Plus Only).

For all its sophistication, however, a major benefit of WordPro Plus continues to be its "turn key" design and ease of use. Over 4,000 copies of WordPro are currently in use worldwide and are available to Dealers and International Distributors exclusively through Professional Software Inc.

Professional Software also announced that its WordPro Plus programs provide capability for multi-user word processing. Together with the Multi-Cluster, WordPro 3 Plus and 4 Plus allow the use of up to 8 CPU's with one disk drive, thereby creating a real-time "Multi-User" word processing system.

In addition to other forms of dealer support, Professional Software has introduced its exclusive Dealer Demo Pak. This Demo Pak contains demonstration programs of WordPro 3 Plus and WordPro 4 Plus, a Commercial Systems Sales Training Manual with emphasis on word processing sales, four-color WordPro Plus Posters, and an extremely complete WordPro Literature Package designed to give dealers concise and professional marketing support.

For more information contact:
Professional Software Inc.,
166 Crescent Road, Needham, MA 02194. (617) 444-5224 Telex 951579

June Workshops Announced

Virginia Polytechnic Institute And State University has announced a set of workshops. The programs will be directed by Dr. Paul Field, Dr. Chris Titus, Dr. Jon Titus, and Mr. David Larsen. These courses will be on the Virginia Tech campus in Blacksburg, Virginia:

1. Digital Electronics for Automaton and Instrumentation, June 22, 23, 1981.
2. Microcomputer Design Interfacing, Programming, and Application using the Z80/8085/8080. June 24, 25, 26, 27, 1981.

All workshops are hands on with the participant designing and testing concepts with the actual hardware. For more information, contact Dr. Linda Leffel, C.E.C., Virginia Tech, Blacksburg, VA 24061. (703-961-5241).

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

SYBEX Announces The Pascal Handbook

SYBEX has announced the release of THE PASCAL HANDBOOK, By Jacques Tiberghien ($14.95). Described as a comprehensive dictionary of all features for most existing versions of Pascal, THE PASCAL HANDBOOK contains every symbol, reserved word, identifier and operator for UCSD, Jensen-Wirth, OMSI, Pascal Z, HP1000, ISO and CDC Pascals.

Arranged in alphabetical order, each of the over 180 entries contains the definition, syntax diagram, semantic description, implementation details, and program examples in a format structured for ease in accessibility and application.

This unique book has been designed to facilitate the use of Pascal for everyone: For the novice, it provides immediate access to definitions and examples. For the experienced programmer it is a single source of comprehensive information.

For more information, contact:
SYBEX, INC.
2344 Sixth Street
Berkeley, California 94710
415/848-8233

Clock/Calendar Feature Dropped From Apple III Personal Computers

CUPERTINO, CA — February 10, 1981 — Apple Computer Inc. said today it would no longer offer a special built-in clock/calendar circuit as part of the Apple III personal computer. As a result, the price of the Apple III has been reduced by $50.

The battery operated integrated circuit is not critical to the Apple III's operation. It is used to log time and date information automatically on files the computer has stored. Users, typically those keeping accounting records, can enter this information manually from the computer keyboard.

"We are removing the clock chip from the Apple III computer because we have not been able to obtain a supplier that can meet Apple's rigid quality and reliability standards," said Barry N. Yarkoni, Apple III product marketing manager. "We feel that elimination of this circuit will have a positive effect on Apple III manufacturing schedules," Yarkoni added.

Customers who currently have Apple IIIs will receive a $50 rebate from the company. Letters announcing the change and offering the rebate have been mailed to all Apple III owners who have returned a warranty card. Dealers have also been notified.

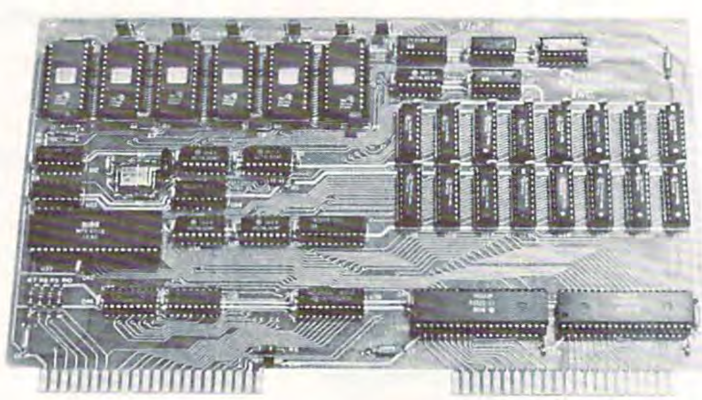
Customer deliveries of the Apple III computer began in November 1980. New U.S. prices start at $4,190.

New VP Marketing At Atari's Computer Division

Sunnyvale, CA — February 2 -Rigdon Currie has joined Atari, Inc. as vice president of marketing for the Computer Division, the announcement made by Raymond E. Kassar, Atari chief executive officer and chairman of the board. In the newly created position, Currie will be responsible for marketing and sales of the Atari 400™ and 800™ personal computers, including peripherals and software.

"We are very pleased to welcome Currie aboard," said Kassar. "He comes to Atari at a time when his marketing experience, both domestic and international, will be particularly valuable. The demand for Atari personal computer systems has

New 6502-Based Single Board Computer



The CPU 65/08 SBC for general purpose industrial and commercial applications is available from Systems Innovations, Inc., Lowell, Mass. Utilizing the popular 6502 Microprocessor, the board will accomodate up to 24K of ROM/EPROM and 8K of RAM.

Two on board VIA's provide 40 I/O lines including 4, 16 bit timer/counters, two serial lines and 14 levels of interrupt. The I/O buss supports DMA and is fully buffered with pinouts equivalent to the KIM 4 standard, thereby allowing the CPU to drive expansion boards directly.

In small quantities, the unit is priced at $275.00 each and is available off the shelf. For further information contact: Systems Innovations, Inc., N.R. Prevett, P.O. Box 2066, 505 Westford Street, Lowell, Mass., 01851

Big Savings On Atari & PET!

No Risk - No Deposit On Phone Orders - Shipped Same Day You Call° - C.O.D. or Credit Card

* On all in stock units

Please Call Between 11AM & 6PM (Eastern Standard Time)

(800) 233-8950

ATARI® 800™ PERSONAL COMPUTER

List $1080

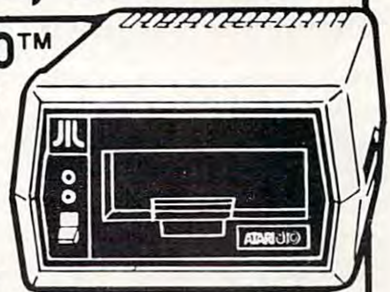
$759

ATARI®

ATARI® 810™ DISC DRIVE

List $599.95

New Low Price only $489.00



Maxell Disks	10 for $34
Sycom Disks	10 for 29
Atari Disks	5 for 22

400 8K	$419
400 16K	449
410 Recorder	62
815 Disk	1199
822 Printer	359
825 Printer	779
830 Modem	159
850 Interface Module	179
CX852 8K RAM	94
CX853 16K RAM	149
CX70 Light Pen	64
CX30 Paddle	18
CX40 Joystick	18
CX86 Printer Cable	42
CO16345 822 Thermal Printer Paper	5
CAO16087 825 80-col. Printer Ribbon (3/box)	17
CX4119 Conversational French	45
CX4118 Conversational German	45
CX4120 Conversational Spanish	45
CX4125 Conversational Italian	45
CXL4009 Chess	30
CXL4011 Star Raiders™	45
CXL4004 Basketball	30
CXL4006 Super Breakout™	30
CXL4010 3-D Tic-Tac-Toe	30
CXL4005 Video Easel™	30
CXL4007 Music Composer	45
CXL4015 TeleLink™	20
CXL4002 BASIC Computing Language	46
CXL4001 Education System Master	21
CXL4003 Assembler Editor	45

CX4115 Mortgage & Loan Analysis	$13
CX4101 An Invitation to Programming 1	17
CX4106 An Invitation to Programming 2	20*
CX4117 An Invitation to Programming 3	20*
CX4107 Biorhythm	13
CX4103 Statistics I	17*
CX4121 Energy Czar	13
CX4108 Hangman	13
CX4102 Kingdom	13
CX4112 States & Capitals	13
CX4114 European Countries & Captials	13
CX4105 Blackjack	13
CX4111 Space Invaders	18

* These are scheduled for release in the first quarter of 1981.

Talk & Teach Courseware;
CX6001 thru CX6017 23

Atari Specialists . . . We carry it all!

Combination Special!

825 Printer & 850 Interface

825 Lists for $1000
Regular Mail Order Price: $779
850 Lists for $220.
Regular Mail Order Price: $179

Save $20.00

Buy Both For Only $938

CX8104 Atari 810 Master Diskette II

New DOS 2 Operating System Master

only $21.00

Microtek RAM 16K or 32K

- Full 1 year warranty
- Compatible with 400 or 800
- Assembled and Tested

16K	$ 99
32K	189

Atari March Releases:

CX8106 Bond Analysis	$20
CX8107 Stock Analysis	20
CX8108 Stock Charting	20
CX4104 Mailing List	17
CX4110 Touch Typing	20
CX8102 Calculator	24
CX4109 Graph It	17
CX4120 Conversational Spanish	45

Software

WordPro 3 (40 col.)	$186
WordPro 4 (80 col.)	279
WordPro 4 Plus (80 col.)	339
Visicalc - Apple	$122
Atari	163
PET	163

commodore

Commodore Computers:

4032 N	$1090
4032 B	1090
8032	1499

Commodore Peripherals:

CBM 2022 Printer	675
CBM 4040 Dual Drive Floppy Disk	1090
CBM 8050 Dual Drive Floppy Disk	1420
CBM C2N Cassette Drive	87
Tally 8024 Printer	1679

Commodore Software

EBS Accounts Receivable Inventory System	$695
Dr. Daley Mailing List	129
Dr. Daley Inventory	89
OZZ Information System	329
BPI General Ledger	329
Tax Package	399
Dow Jones Portfolio Management	129
Pascal	239

Cables:

PET to IEEE Cable	37
IEEE to IEEE Cable	46

Printers

NEC 5530	$2495
Diablo 630	2195
Trendcom 100	299
Trendcom 200	489
Paper Tiger 445G	769
Paper Tiger 460G	1219
Epson MX-80	539

To Order:

Phone orders invited (800 number is for order desk only). Or send check or money order. Equipment Shipped UPS collect. Pennsylvania residents add 6% sales tax. Add 3% for Visa or MC. Equipment is subject to price change and availability without notice.

Computer Mail Order
501 E. Third St.
Williamsport, PA 17701
(717) 323-7921

been rising dramatically. Currie's knowledge of worldwide computer marketing should facilitate our sales operation and allow the Computer Division to expand rapidly."

Currie will be reporting to Roger Badertscher, president of the Computer Division.

Previously, Currie spent more than 16 years with the Xerox Corporation in various managerial positions on the corporate staff and with Xerox Data Systems. For the past 3½ years he was vice president of marketing and planning for Diablo Systems, a Xerox subsidiary. While there, he led Diablo to increased sales of word-processing, small business and terminal products in the OEM and retail markets.

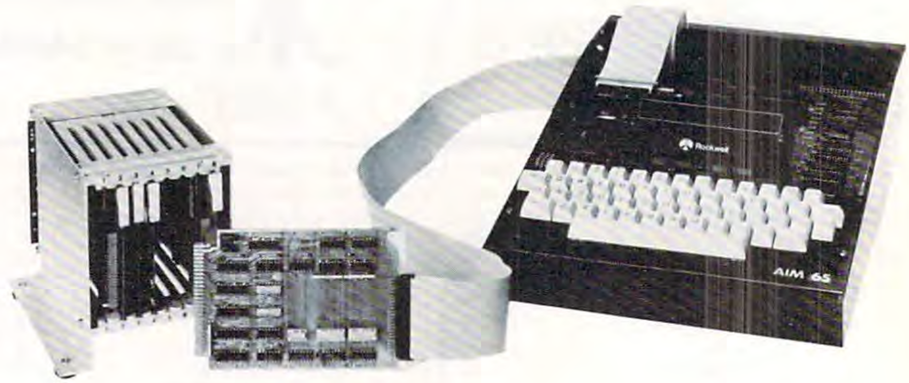
A native of Atlanta, Georgia, Currie received a bachelor's degree in industrial engineering from Georgia Institute of Technology in 1951, and received a master's degree in business administration from Harvard University in 1956.

Currie and his family reside in Palo Alto, California.

STD BUS Expansion for the AIM 65 Computer

As part of its AIM-Mate series of expansion products for the AIM 65, Forethought Products has released its STD-Mate* interface to the STD BUS. By directly substituting for the STD BUS processor card, STD-Mate allows full speed use of STD BUS cards by the AIM 65 both for expansion of existing AIM 65 systems and development/debugging of stand alone STD BUS systems. STD-Mate is available off-the-shelf for $140. Forethought Products, 87070 Dukhobar Rd., Eugene, OR 97402; (503) 485-8575

*TM Forethought Products



Atari Memory Boards

Microtek Peripherals Corporation has announced the release of two memory boards for the ATARI 400/800 computers. The AT-16, a 16k 200ns board, and the AT-32, a 32k 200ns board are designed to be completely compatible with all existing hardware and software, and are user-installable with no modifications. Both products carry MICROTEK's standard one year warranty. Retail pricing is $119.50 on the AT-16 and $199.50 on the AT-32.

Additional products currently available include cables and board extenders. MICROTEK PERIPHERALS plans to introduce a number of ATARI peripherals in the coming months. For further information contact: MICROTEK PERIPHERALS CORPORATION, 9514 Chesapeake Drive, San Diego, CA 92123. 800-854-1081 (in CA 714-278-0630)

Educational Software Announcement

Teacher's Pet offers a listing of over 20 original PET programs for intermediate grade students in math, language arts, and logic. These programs are written by Glenn Fisher, an experienced teacher and programmer who has published several articles on programming techniques. Most programs are drill and practice, with graphics and scoring. They have all been thoroughly tested in classrooms. Complexity ranges from "Times," a simple multiplication facts drill, to "Decimal X," a tutorial which takes a student step by step through decimal multiplication problems at the PET keyboard and allows the teacher to set the size of the problem and the number of decimal places.

Language programs vary from "Parts of Speech," a drill in recognizing word use, to "Comma," a program written so that inexperienced computer-users can change the data easily. Programs offered include a grading program and a program to handle California state enrollment data for elementary school.

All programs have input protected against careless users so they will run without problems in the classroom. Several programs allow 1 to 4 students to enter their names and be scored separately. To request the list of programs, write: Teacher's Pet, Dept. C, Glenn Fisher, 1517 Holly St., Berkeley, CA 94703

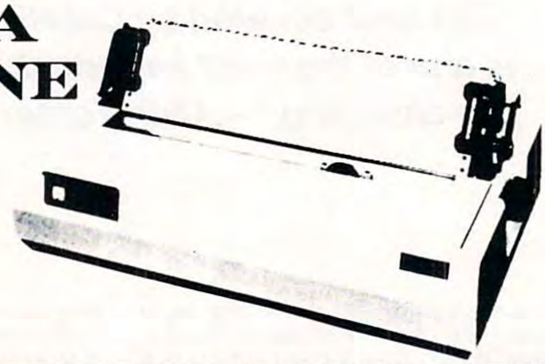
WEST COAST GRAND OPENING!!!

ΩMEGA SALES COMPANY

CALL TOLL FREE 1-800-235-3581

GRAND OPENING
SPECIAL OF THE MONTH!!

OKIDATA MICROLINE 83



$929

DIABLO 630

$1995.00

(TRACTOR $200)

NEC SPINWRITER 5510-5530

$2395.00

(TRACTOR $200)

APPLE II+
48K-$1150.00

C. ITOH STARWRITER 25
PARALLEL OR SERIAL
$1495.00

APPLE DISK DRIVE
DOS 3.3 w/CONTROLLER
$545.00

OKIDATA MICROLINE-80	BASE II	CBM	HAZELTINE 1420
$469.00	$649.00	CALL FOR PRICE	$799

1. TWO LOCATIONS TO SERVE YOU BETTER.
2. NO SURCHARGE FOR CREDIT CARDS.
3. WE ACCEPT C.O.D.'S.

ΩMEGA SALES CO - WEST COAST - 3353 OLD CONEJO RD #102
NEWBURY PARK CA 91320 TEL 1-805-499-3678

EAST COAST OFFICE

ΩMEGA SALES COMPANY

CALL TOLL FREE 1-800-556-7586

2ND SPECIAL OF THE MONTH!!

EPSON MX-80 PRINTER

$479

INTERFACES: APPLE INTERFACE & CABLE $90, RS*232-$70, IEEE $55, TRS*80 CABLE $35

ATARI 800
$759.00

INTERTEC SUPERBRAIN
32K - $2449.00
64K - $2649.00
QD - $3195.00

NORTHSTAR HORIZON II
32K QUAD. DENS.
$2975.00

NEC MONITOR $219.00

ATARI 400
8K - $399.00
16K - $499.00

RADIO SHACK
16K LEVEL II
MODEL 3
$875.00

RADIO SHACK
64K MODEL 2
$3449.00

ANADEX
DP-9500
$1249.00

SOROC
120 - $630
140 - $949

TELEVIDEO 912C - $679.00, 920C - $739.00

4. Stock shipments shipped usually same day or next day. Limited quantities available.

5. All equipment factory fresh with manufacturer's warr.

6. No hidden charges - We live by our published prices.

ΩMEGA SALES CO - EAST COAST - 12 MEETING ST.
CUMBERLAND, RI 02864 TEL 1-401-722-1027

VISA

PRICES SUBJECT TO CHANGE WITHOUT NOTICE.

CAPUTE!

Corrections/Clarifications

In our three part series by Hal Chamberlin, **"Expanding KIM-Style 6502 Single Board Computers"**, we neglected to point out that Mr. Chamberlin is Vice President of Research And Development for Micro Technology Unlimited. You may write him at P.O. Box 12106, Raleigh, NC 27605.

* * * * * * * * * * * * *

Our February, 1981 issue carried an article by Bruce Land entitled, **"A Terminal For 'KAOS' (KIM, Aim, OSI, Sym)"**. In evaluating the Netronics terminal, Bruce said:

"The Netronics 20 ma current loop is not isolated like the Xitrex, and so may not work well with some devices. It does not work well with all the devices I have tried including KAOS systems." (p. 132, column 2)

At least that's what we printed. What Bruce really wrote was:

"It does work well with all the devices I have tried..."

* * * * * * * * * * * * * *

"The 25¢ Apple II Real Time Clock" (February, 1981; page 72, figure 3)

An error in illustration inadvertantly left a tie that "shouldn't bind". Here are both the incorrect and corrected versions:

"Ticker Tape Atari Messages", (February, 1981, page 75)

As you've noticed, there was a problem with the program. Here are the fixes:

1. The [ ]'s used in the listing are ( )'s. Nobody we talked to had a problem with this, but we thought we'd mention it.

2. In line 20, B is set equal to a blank; this should be B$.

3. Add a line 16 that sets Y$ = W$.

4. Finally, the contents of W$ and Y$ determine your "moving borders"; if you put 20 spaces in W$, you won't see anything. If you put in * alternating with spaces, then you'll see what we mean.

Play around with it, and compose your own borders.

* * * * * * * * * * * * * *

Program Listings for COMPUTE

Cursor control characters will appear in source listings as shown below:

```
h=HOME            , ĥ=CLEAR SCREEN
↓=DOWN CURSOR     , ↑=UP CURSOR
→=RIGHT CURSOR,     ←=LEFT CURSOR
r=REVERSE         , r̂=REVERSE OFF
```

Graphics (i.e. shifted) characters will appear as the unshifted alphanumeric character with an underline. This does not apply to the cursor control characters. The Spinwriter thimble doesn't have a backarrow symbol, so a "~" is used instead.

The "¬" is used to indicate the beginning of a continuation line. It is also used to indicate the end of a line which ends with a space. This prevents any spaces from being hidden.

FIGURE 3

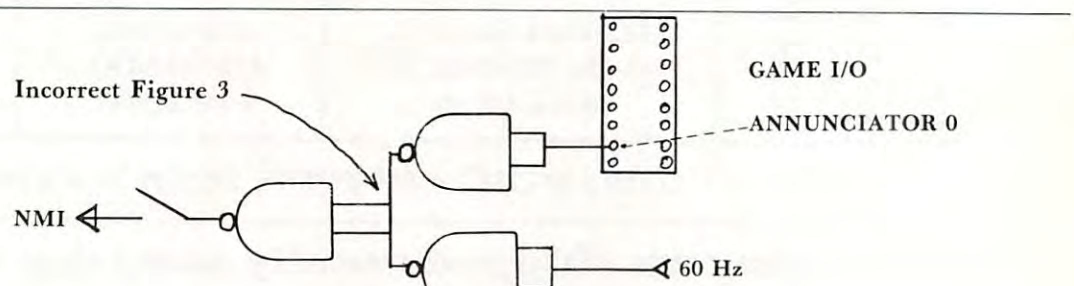
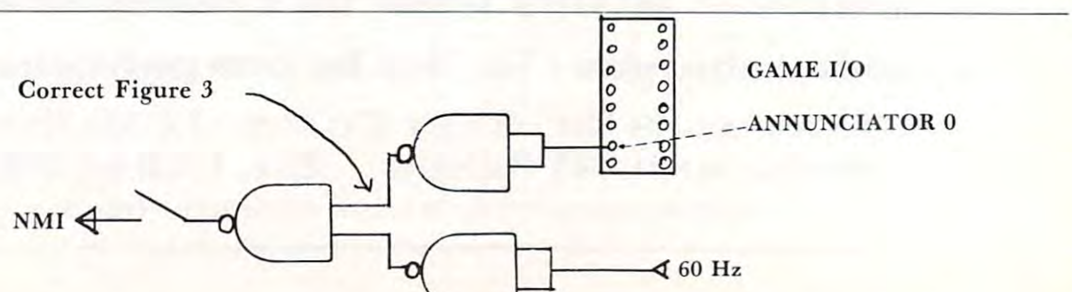
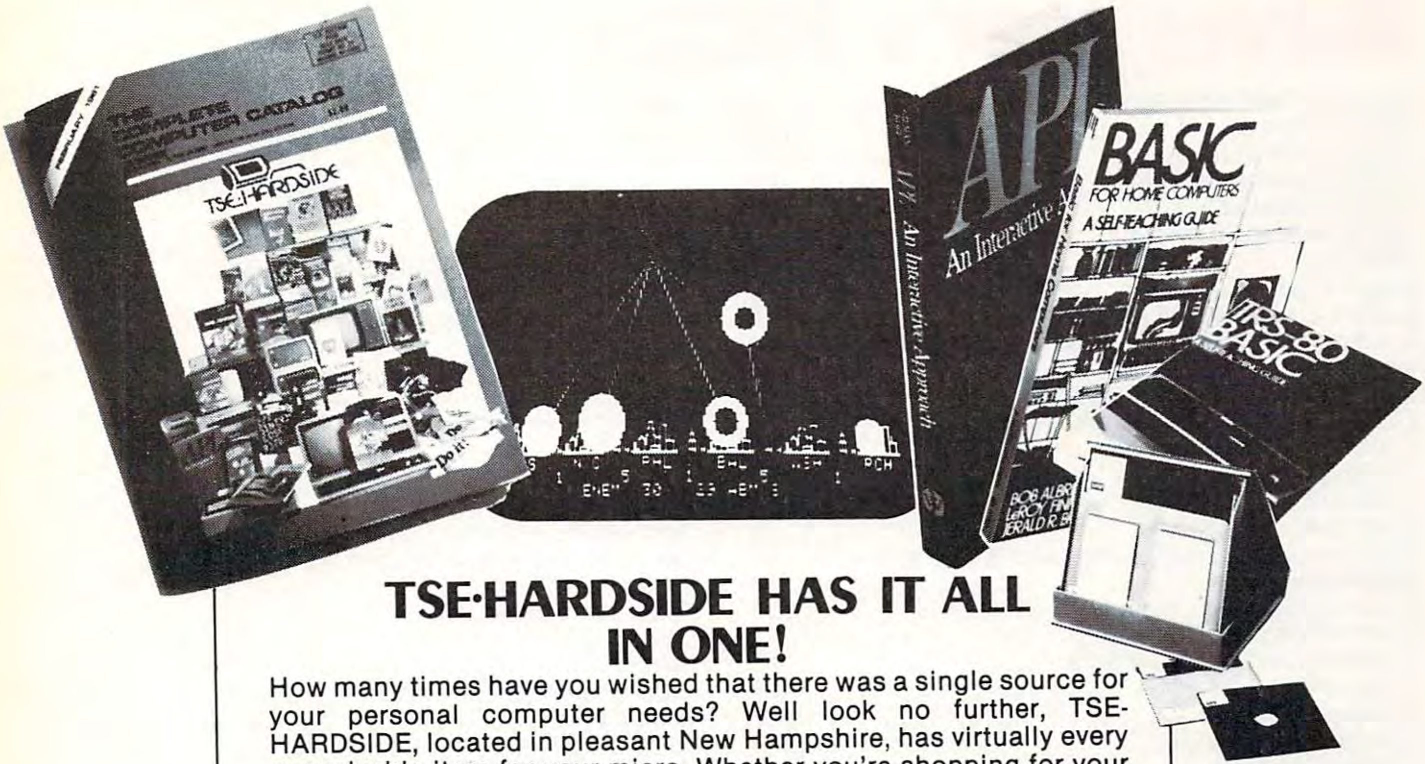


FIGURE 3

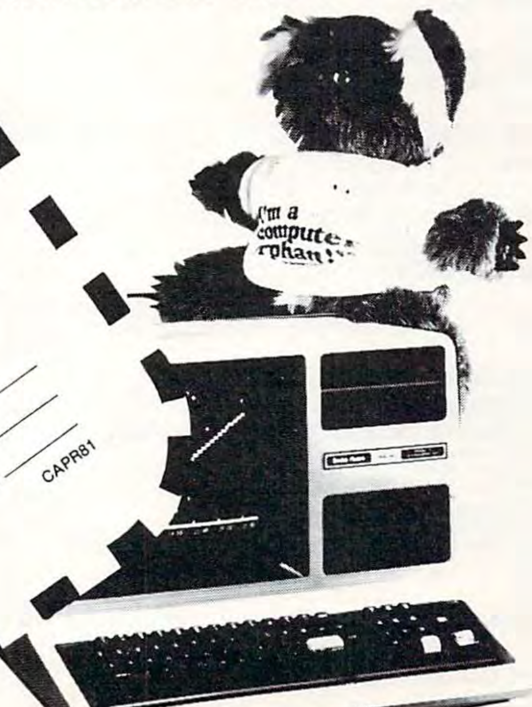




TSE·HARDSIDE HAS IT ALL IN ONE!

How many times have you wished that there was a single source for your personal computer needs? Well look no further, TSE-HARDSIDE, located in pleasant New Hampshire, has virtually every conceivable item for your micro. Whether you're shopping for your Apple, Pet, TRS-80 or Atari, TSE-HARDSIDE has it all. We stock hardware, software, books, magazines and specialty items for all of the popular machines. So the next time you're out shopping for your system don't be surprised, be satisfied. Remember the name TSE-HARDSIDE as your choice for quality, service and reliability.

TSE·HARDSIDE
6 South St., Milford, NH 03055 (603) 673-5144
TOLL FREE OUT-OF-STATE 1-800-258-1790



Save $4.00

If you or a friend would like to subscribe to our TSE/HARDSIDE Complete Computer Catalog, simply send $2.00 and receive a $4.00 certificate good toward your first purchase of software, hardware, books or specialty items from TSE/HARDSIDE. I have a (check one) ☐ TRS-80, ☐ APPLE ☐ ATARI ☐ PET. A photo-copy of this coupon will be accepted.

Name ______________________

Address ______________________

City ____________ State ________ Zip ________ CAPR81

MAIL TO: TSE/HARDSIDE
6 South Street
Milford, NH 03055

Advertiser's Index

The 6502 Resource Magazine
COMPUTE! PET•ATARI•APPLE OSI•KIM•SYM•AIM

My computer is:
☐ **PET** ☐ **APPLE** ☐ **ATARI** ☐ **OSI**
☐ **KIM** ☐ **SYM** ☐ **AIM** ☐ **OTHER**__________
☐ **Don't yet have one...**

During the next year I expect to buy:
☐ **computer** ☐ **printer**
☐ **disk drive** ☐ **other peripherals**

☐ Please enter my 1 year (12 issue) subscription to COMPUTE!
☐ New subscription ☐ Renewal subscription
☐ $20.00 U.S. subscription
☐ $25.00 Canada and international surface mail: Payment in U.S. funds, drawn on U.S. bank or money order.
☐ Bill me (U.S. only). $1.00 billing fee.

Name, Address: ____________________

Charge my: ☐ Visa ☐ MC
Number________________ Expires_____/_____/

4 5 6 7 8 9 10 11 12

The Editor's Feedback:

My computer is:
☐ **PET** ☐ **APPLE** ☐ **ATARI** ☐ **OSI**
☐ **KIM** ☐ **SYM** ☐ **AIM** ☐ **OTHER**__________
☐ **Don't yet have one...**

During the next year I expect to buy:
☐ **computer** ☐ **printer**
☐ **disk drive** ☐ **other peripherals**

Content:
Best Article This Issue (page #, title)

____

Other suggestions:

4 5 6 7 8 9 10 11 12

The 6502 Resource Magazine
COMPUTE! PET•ATARI•APPLE OSI•KIM•SYM•AIM

My computer is:
☐ **PET** ☐ **APPLE** ☐ **ATARI** ☐ **OSI**
☐ **KIM** ☐ **SYM** ☐ **AIM** ☐ **OTHER**__________
☐ **Don't yet have one...**

During the next year I expect to buy:
☐ **computer** ☐ **printer**
☐ **disk drive** ☐ **other peripherals**

☐ Please enter my 1 year (12 issue) subscription to COMPUTE!
☐ New subscription ☐ Renewal subscription
☐ $20.00 U.S. subscription
☐ $25.00 Canada and international surface mail: Payment in U.S. funds, drawn on U.S. bank or money order.
☐ Bill me (U.S. only). $1.00 billing fee.

Name, Address: ____________________

Charge my: ☐ Visa ☐ MC
Number________________ Expires_____/_____/

4 5 6 7 8 9 10 11 12

Place
Stamp
Here

COMPUTE.
Post Office Box 5406
Greensboro, NC 27403

Place
Stamp
Here

COMPUTE!
Post Office Box 5406
Greensboro, NC 27403

Place
Stamp
Here

COMPUTE!
Post Office Box 5406
Greensboro, NC 27403

The Great American Solution Machine.

Meet Commodore. The business computer that's providing solutions for more than 100,000 people all over the world. Built by one of the pioneers in office machines. With a reputation for quality that can only come from vertical integration of the total manufacturing process. Commodore builds, not assembles.

Compare Commodore's word and data processing capabilities with computers costing twice or even three times as much. You'll see why so many small businesses are turning to Commodore for solutions to problems as varied as these:

☐ A car leasing company's customers were terminating too early for account profitability. Solution: A 16K Commodore. It analyzes cash flow on over 1200 accounts, identifies those for early penalties, and even writes up lease contracts. Commodore paid for itself within weeks.

☐ A law firm needed a high quality, easy-to-use, affordable word processing system. Solution: Commodore plus its WordPro software package. At a $6,000 savings.

☐ A gasoline retailer needed to inventory, order and set prices; determine Federal and state income taxes; and comply with Federal pricing and allocation regulations. All done daily, weekly, monthly and yearly. Solution: Commodore. It keeps his business on track—and Uncle Sam off his back.

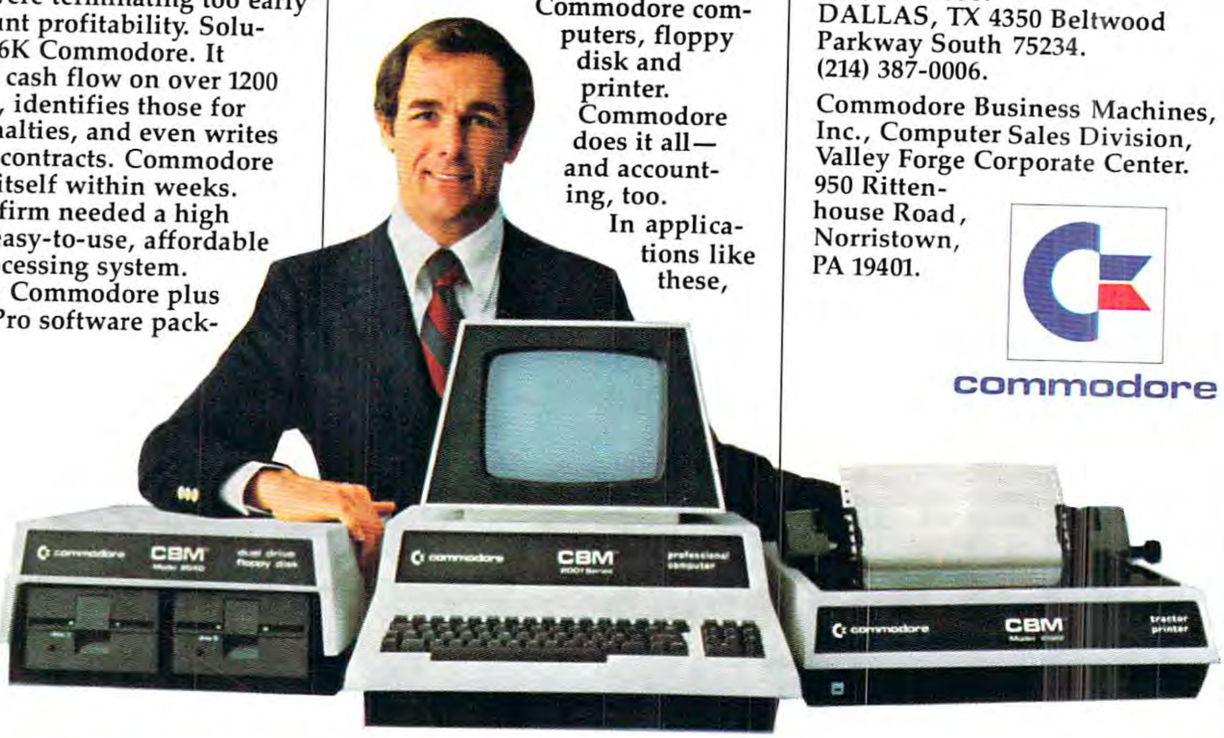
☐ A paint and wallpaper store had to inventory over 600 expensive wallpaper lines for profitability, monitor distributor sales, set and track salesmen's goals, and help the customer select the right size, pattern and quantity. Solution: Two 32K Commodore computers, floppy disk and printer. Commodore does it all—and accounting, too.

In applications like these, and many more, Commodore solves the problems that stand in the way of increased profitability. Commodore can provide the solution in your Great American business, too. Find out more by calling or writing any of Commodore's District Sales Offices.

COSTA MESA, CA 2955 N. Airway Avenue 92626. (714) 979-6307.
SANTA CLARA, CA 3330 Scott Boulevard 95051. (408) 727-1130.
DECATUR, GA 5360 Snapfinger Woods Drive 30035. (404) 987-3311.
BENSENVILLE, IL 790 Maple Lane 60106. (312) 595-5990.
NORRISTOWN, PA 950 Rittenhouse Road 19401. (215) 666-7950.
DALLAS, TX 4350 Beltwood Parkway South 75234. (214) 387-0006.

Commodore Business Machines, Inc., Computer Sales Division, Valley Forge Corporate Center. 950 Rittenhouse Road, Norristown, PA 19401.

commodore



COMMODORE